

# Impressum

Copyright (C) LunetIX 1997

Die Hypertextversion des Linux Anwenderhandbuches ist ein urheberrechtlich geschütztes Werk der Autoren Sebastian Hetze, Dirk Hohndel, Olaf Kirch und Martin Müller.

Das ausschließliche Recht auf die Verbreitung des Werkes auf elektronischen Medien liegt bei LunetIX, Müller & Hetze GbR; Donaustr. 16, 12043 Berlin.

Das vorliegende OnLine-Dokument basiert auf der 7. Auflage des Linux Anwenderhandbuches von LunetIX. Die gedruckte Version ist unter der ISBN 3-929764-06-7 in jeder besseren Buchhandlung erhältlich.

Das vorliegende Dokument ist mit dem Programm latex2html von Nikos Drakos aus den aktuellen Sourcen des Linux Anwenderhandbuches erzeugt worden. Einige Abschnitte des Buches sind nicht in der OnLine-Version enthalten. Andererseits sind auch Teile der OnLine-Version aus Platzgründen nicht im gedruckten Buch enthalten.

Wir arbeiten an der Verbesserung und an der Erweiterung sowohl des Linuxhandbuches im Allgemeinen als auch an der Hypertextversion im Speziellen. Wenn Sie Fehler finden und Anregungen oder Kritik zu dem Werk haben, freuen wir uns über eine Mitteilung von Ihnen!

UNIX ist ein eingetragenes Warenzeichen von Univel.

MS-DOS, Windows und Microsoft sind eingetragene Warenzeichen der Microsoft Corporation.

DEC und PDP sind Warenzeichen der Digital Equipment Corporation.

XFree86 ist ein Warenzeichen von The XFree86 Project, Inc.

LunetIX ist ein eingetragenes Warenzeichen von S. Hetze und M. Müller GbR.

Viele weitere Produktbezeichnungen sind Warenzeichen der jeweiligen Hersteller und entsprechend zu behandeln.

## Subsections

- [Foreword by Linux Torvalds](#)
  - [Vorwort zur fünften Auflage](#)
  - [Danksagung](#)
  - [Vorwort zur ersten Auflage](#)
- 

# Vorworte

## Foreword by Linux Torvalds

After being threatened by my old friend Dirk, I find myself writing a new pre-face for the new book.

It's about a year since I released 1.0, and now 1.2 is just about being done. In the meanwhile, many linux things have changed, so it's only fitting that a new release of a linux book would also come out..

Since I last sat down and tried to come up with a foreword, both the linux kernel and the programs around it have been updated many times, and linux now works on both i386 and 68k machines, and work is in progress to port it to the DEC Alpha, MIPS, Sparc and PowerPC architectures. I'm personally happily hacking away at the alpha port, and I think that the kernel has come a long way in the year past.

The Linux community has similarly also developed in the last year, and Linux now has a much broader recognition and a larger following. Nobody thought it would happen quite this way, but hey, I'm not complaining. In fact, I'm smugly rubbing my hands and laughing madly. People around me are giving me worried looks..

This book will cover some of the new kernel things, but will also get you updated on the new XFree86 release, for example. I still hope you'll find this book useful and enjoy it as much as I enjoy it.

Linus Torvalds  
Helsinki, 25.2.95

## Vorwort zur fünften Auflage

Es ist gerade ein halbes Jahr seit Erscheinen der vierten Auflage vergangen, trotzdem ist eine Neuauflage unseres Linux-Anwenderhandbuches notwendig. Mit dieser neuerlichen Aktualisierung und Erweiterung unseres Werkes reagieren wir auf die ungebremst dynamische Entwicklung des Betriebssystems und der Programme, die darauf laufen.

Genau ein Jahr nach Fertigstellung der Linux-Version 1.0 ist jetzt die nächste Major-Release erfolgt: Linux 1.2 ist freigegeben. Damit ist ein neuer Meilenstein in der Entwicklung dieses faszinierend leistungsfähigen Betriebssystems gesetzt. Auf der i386-Basis ist Linux stabil und gründlich getestet,

ein hervorragendes System zum Betrieb und zur Entwicklung von Anwendungen aller Art.

Ebenfalls erst vor einigen Wochen herausgekommen ist die Version 3.1.1 des XFree86 X Window Systems für Linux. Außer einigen Bugfixes bringt sie die Unterstützung weiterer Grafikkarten.

Neue Versionen der Linux-Utilities und der Standardbibliothek des C-Compilers bringen weitere kleinere Neuerungen, wie zum Beispiel die Unterstützung von Locales und NLS.

All diese Entwicklungen sind in der fünften Auflage berücksichtigt.

Wenn Sie inhaltliche Fehler finden oder Kritik und Anregungen zu unserem Buch haben, würden wir uns sehr freuen, von Ihnen zu hören. Diese Rückkopplung ist wohl Wunsch eines jeden Autors - für die weitere Arbeit an dem Linux-Anwenderhandbuch ist sie dringend notwendig.

## Danksagung

Die vorliegende fünfte Auflage des Linux Anwenderhandbuches ist nicht das Produkt der Autoren allein - an dem Gelingen des Werkes haben viele Menschen direkt oder indirekt mitgewirkt, denen wir hiermit herzlich danken wollen.

Zuerst danken wir allen Entwicklerinnen und Entwicklern Freier Software für ihre großartigen Produkte. Die Faszination der „GNU Generation“ und der Wunsch, selbst aktiver Teil davon zu sein, ist dauernde Motivation für unsere Arbeit. Insbesondere gilt unser Dank Linus Torvalds, der uns das unerschöpfliche Thema unseres Buches liefert.

Den Mitarbeiterinnen und Mitarbeitern der Firma J.F. Lehmanns, besonders Bernd Sommerfeld, danken wir für ihre Unterstützung bei der Vermarktung des Buches und für die dauernde Ermutigung zu unserer Arbeit. Ohne sie hätte es das Handbuch vielleicht nie gegeben.

Allen Leserinnen und Lesern, die uns geschrieben haben, danken wir für Lob, Anregungen und Kritik.

Denjenigen, die unser Buch gekauft haben oder es jetzt kaufen werden, danken wir für die finanzielle Unterstützung unseres Projektes.

Wir danken Sven Schüle, Ralf Flaxa, Stefan Probst und Michael Wiedmann für ihre Anregungen und für die Unterstützung auf dem Weg vom Manuskript zum fertigen Buch.

Katja Lachmann (na194@fim.uni-erlangen.de) danken wir für die Übersetzung der GPL ins Deutsche, die uns als Grundlage für den Anhang F gedient hat.

Nur durch die Summe der Einzelleistungen ist es möglich, daß dieses Handbuch weiterhin im Selbstverlag herausgegeben werden kann.

Berlin, Dulles, Darmstadt, den 9.3.1995

Sebastian Hetzes (she@lunetix.de)  
Dirk Hohndel (hohndel@lunetix.de)  
Olaf Kirch (okir@lunetix.de)  
Martin Müller (mm@lunetix.de)

# Vorwort zur ersten Auflage

Wer heute einen „kleinen“ Computer kauft, privat für Zuhause oder als Arbeitsplatzrechner für allgemeine Aufgaben, der kauft Technik mit einem Leistungspotential, das noch zu Beginn der achtziger Jahre den Großrechnern der Konzerne vorbehalten war. Standardmäßig werden diese PC's noch immer mit dem aus dem Jahre 1981 stammenden MS-DOS betrieben. Die damit verbundenen Einschränkungen sind langsam zu Fesseln geworden, von denen sich viele Benutzer befreien wollen. Kein Wunder also, daß der Kampf der Softwaregiganten um die Nachfolge von MS-DOS in vollem Gange ist.

Als Außenseiter in dieser Konkurrenz um Profit und Marktanteile tritt Linux auf, ein Nachbau des altbewährten Unix Betriebssystems. Linux ist freie Software, es steht unter der GNU General Public License. Diese Lizenz garantiert jedem Benutzer das Recht, das Programm beliebig oft zu kopieren, sowie den freien Zugang zu den Quelltexten.

Als Unix-Clone ist Linux eine vollwertige Basis für die meisten Unix-Programme. Der gcc C-Compiler der Free Software Foundation ist der Schlüssel zum gesamten Angebot an freier Unix Software. Durch diesen Rückgriff auf den bestehenden Pool freier Software ist ein „Basissystem“ von Linux heute ebenso vielseitig und umfangreich wie die „professionelle“ Konkurrenz.

Etwas, was Linux wirklich fehlt, sind die Handbücher, die normalerweise zum Lieferumfang eines Betriebssystems gehören. Es gibt die englischen Manualpages und eine Reihe weiterer ebenfalls in Englisch abgefaßter Hilfstexte, die online gelesen werden können. Außerdem wird von der internationalen Linuxgemeinde an einem Dokumentationsprojekt gearbeitet. Um eine Lücke zu schließen, die unserer Meinung nach bei Unix-Einsteigern vor allem auch wegen der Sprachbarriere entsteht, haben wir dieses Buch geschrieben.

Wir stellen uns vor, daß der Leser dieses Buches bereits grundlegende Fertigkeiten im Umgang mit Computern besitzt, die er beispielsweise unter MS-DOS erworben hat. Weiter gehen wir davon aus, daß der Leser keine oder nur wenige Kenntnisse von Unix hat, und daß er kein allgemeines Unix-Buch im Regal stehen hat.

Wir wollen einen Anfänger in die Lage versetzen, sein Linux Basissystem zu konfigurieren, mit dem Dateisystem umzugehen, Benutzer und Benutzergruppen zu verwalten. Und wir wollen dazu ein wenig Hintergrundinformation zum Betriebssystem geben. Wir haben nicht den Anspruch die Tiefen von Linux zu ergründen. Das halten wir für eine Überforderung des Lesers; abgesehen davon erschien uns der Aufwand dafür zu groß. An einem derartigen Projekt wird, wie bereits gesagt, international gearbeitet, und der ernsthaft ambitionierte Leser wird früher oder später um die Lektüre der englischen Originaltexte nicht herumkommen.

Berlin, den 1.3.1993

*LunetIX Softfair*

Martin Müller &

Wir nehmen Stellung:

## Gegen Faschismus und Rassismus!

Diese Ideologien verdrängen die inneren Probleme unserer Gesellschaft nach außen, anstatt sie zu lösen.

Es gibt keine einfachen Rezepte, schon gar keine nationalistische Lösung. Eine grundlegende Neuorientierung der Gesellschaft ist notwendig. Sie kann aber nur mit den Menschen und Völkern der „anderen“ Länder und Kontinente entwickelt werden, nie gegen sie.

Die sogenannte Asylproblematik wird nicht durch Vertreibung gelöst, sie entsteht durch Vertreibung!

Es gibt viele Arten einen Menschen zu töten.

Nur wenige davon zählen in diesem Land als Asylgrund.

In tiefer Trauer um einen 1992 von stolzen, deutschen Faschisten ermordeten Freund, und um alle anderen Opfer faschistischer und rassistischer Gewalttaten.

Die Dummheit der Täter und das Elend ihres Lebens entschuldigen diese Taten nicht.

Auch 1997, im Europäischen Jahr des Flüchtlings, sind Menschen in Deutschland wegen ihrer Herkunft, ihres Glaubens oder ihres Aussehens Gewalttaten zum Opfer gefallen.

**Schaut nicht weg! Greift ein!**

Berlin, den 3.10.1997

Sebastian Hetze, Dirk Hohndel, Olaf Kirch und Martin Müller

LunetIX

---



# Inhalt

- [Impressum](#)
- [Vorworte](#)
  - [Foreword by Linux Torvalds](#)
  - [Vorwort zur fünften Auflage](#)
  - [Danksagung](#)
  - [Vorwort zur ersten Auflage](#)
- [Inhalt](#)
- [Grundlagen](#)
  - [Geschichte](#)
    - [Steinzeit](#)
    - [Unix](#)
    - [Minix](#)
    - [Linux](#)
  - [Das Betriebssystem](#)
  - [Die ersten Schritte](#)
    - [Verzeichnisse und Dateien](#)
      - [Das aktuelle Verzeichnis](#)
      - [Namen und Pfade](#)
      - [Das Verzeichnis wechseln](#)
      - [Den Inhalt eines Verzeichnisses anzeigen](#)
      - [Eigentum und Zugriffsrechte](#)
      - [Verzeichnisse erstellen und löschen](#)
      - [Dateien erstellen, verschieben und löschen](#)
      - [Dateien anzeigen](#)
    - [Datenströme](#)
      - [Verknüpfung mehrerer Kommandos: Pipelines](#)
    - [Eingabehilfen von der Shell](#)
    - [Virtuelle Terminals und Hintergrundprozesse](#)
  - [Zahlensysteme](#)

- [Reise durch's Dateisystem](#)
  - [Konzepte](#)
    - [Der File-System-Standard](#)
  - [Rootpartition und Wurzelverzeichnis](#)
    - [Das Verzeichnis /boot](#)
    - [Das Verzeichnis lost+found](#)
    - [Das Verzeichnis /mnt](#)
    - [Das Verzeichnis /root](#)
  - [Die Binärverzeichnisse](#)
  - [Die Gerätedateien im Verzeichnis /dev](#)
    - [Der Arbeitsspeicher](#)
      - [Die RAM-Disks /dev/ram\\*](#)
      - [Zufallszahlen aus /dev/random](#)
      - [Die Senke /dev/null](#)
      - [Die Quelle /dev/zero](#)
      - [Die Fehlerquelle /dev/full](#)
      - [Der IO-Bereich /dev/port](#)
      - [Der Arbeitsspeicher /dev/mem und /dev/kmem](#)
    - [Runde Scheiben](#)
      - [Die Diskettenlaufwerke /dev/fd\\*](#)
      - [Die IDE-Festplatten](#)
      - [Die CD-ROM-Laufwerke](#)
    - [Zeichenorientierte Ein- und Ausgabegeräte](#)
      - [Die Console und virtuelle Terminals](#)
      - [Pseudoterminals für die grafische Benutzeroberfläche](#)
      - [Die Busmäuse](#)
      - [Der Wecker: /dev/rtc](#)
      - [Die parallele Druckerschnittstelle /dev/lp?](#)
      - [Die Soundkarte](#)
    - [Daten am laufenden Band](#)
      - [Die QIC-02-Streamer /dev/rmt? und /dev/tape\\*](#)
      - [Die SCSI-Streamer /dev/st? und /dev/nst?](#)
      - [Der Floppystreamer \(ftape\)](#)
  - [Die Konfigurationsdateien im Verzeichnis /etc](#)
    - [/etc/adjtime](#)



- [/etc/fdprm](#)
- [/etc/fstab](#)
- [/etc/gettydefs](#)
- [/etc/group](#)
- [/etc/hosts](#)
- [/etc/inittab](#)
- [/etc/issue](#)
- [/etc/ld.so.conf](#)
- [/etc/login.defs](#)
- [/etc/magic](#)
- [/etc/man.config](#)
- [/etc/motd](#)
- [/etc/nologin](#)
- [/etc/passwd](#)
- [/etc/printcap](#)
- [/etc/profile](#)
- [/etc/psdatabase](#)
- [/etc/rc\\*](#)
- [/etc/securetty](#)
- [/etc/shells](#)
- [/etc/syslogd.conf](#)
- [/etc/termcap](#)
- [Nichts über TCP/IP](#)
- [Home, Sweet Home](#)
  - [Das möblierte Zimmer](#)
- [Die Bibliotheken in `./lib`](#)
  - [Compiler, Bibliotheken und Daten in `/usr/lib`](#)
    - [Der C-Compiler](#)
- [Erweiterungspakete in `/opt`](#)
- [Die Prozeßdaten im Verzeichnis `/proc`](#)
- [Die temporären Dateien im Verzeichnis `./tmp`](#)
- [Das Verzeichnis `/usr`](#)
  - [/usr/X11R6](#)
  - [/usr/dict](#)
  - [/usr/doc](#)

- [/usr/games](#)
  - [/usr/include](#)
  - [/usr/local](#)
  - [/usr/info](#)
  - [/usr/man](#)
  - [/usr/share](#)
  - [/usr/src](#)
  - [/usr/spool](#)
- [Das Verzeichnis /var](#)
- [Von GNU's, Muscheln und anderen Tieren](#)
  - [Intro\(1\) - oder die Erklärung der Erklärung](#)
    - [Syntax:](#)
    - [Beispiel:](#)
    - [Siehe auch:](#)
    - [Die 13 goldenen Regeln für ein gelungenes Kommando](#)
  - [ar](#)
    - [Funktion:](#)
    - [Syntax:](#)
    - [Beschreibung:](#)
    - [Optionen:](#)
    - [Autor:](#)
  - [basename](#)
    - [Funktion:](#)
    - [Syntax:](#)
    - [Beschreibung:](#)
    - [Siehe auch:](#)
      - [Autor:](#)
  - [bash](#)
    - [Funktion](#)
    - [Syntax](#)
    - [Beschreibung](#)
    - [Interaktive Shell und Shellprogrammierung](#)
    - [Der Kommandozeileneditor](#)
    - [Der Kommandozeilenspeicher \(history\)](#)
      - [Der Kommandozeilenspeicher im Editor](#)

- [History im C-Shell-Stil](#)
- [Bezugnahme auf eine frühere Kommandozeile](#)
- [Bezugnahme auf ein Wort einer früheren Kommandozeile](#)
- [Modifikation der bezogenen Kommandozeilen](#)
- [Beispiele für die History-Funktion](#)
- [Anpassung des Kommandozeileneditors](#)
  - [Bedingte Ausführung von `.inputrc`](#)
  - [Beispiel:](#)
  - [Der erweiterte Editor: sekundärer Prompt](#)
- [Interpretation der Kommandozeile](#)
- [Kommentare](#)
- [Der Status](#)
- [Shell Grammatik](#)
  - [Reservierte Wörter](#)
  - [Atome, Wörter, Token](#)
  - [Kommandos - nicht unbedingt einfach](#)
- [Quotierung](#)
- [Ein-/Ausgabe-Umleitung](#)
  - [Eingabeumleitung](#)
  - [Ausgabeumleitung](#)
  - [Anfügen der Ausgabe an eine existierende Datei](#)
  - [Zusammenfassung der Standardausgabe mit der Standardfehlerausgabe](#)
  - [Shellscript-Dokumente](#)
  - [Verdoppelung der Dateikennung](#)
  - [Öffnen einer Datei zum Lesen und Schreiben](#)
- [Pipelines](#)
- [Hintergrundprozesse](#)
- [Listen](#)
  - [Listen mit ; und &](#)
  - [Bedingte Ausführung](#)
- [Gruppen und Kontrollstrukturen: Blöcke, Schleifen, Verzweigungen, Funktionen](#)
- [Parameter](#)
  - [Shell- und Umgebungsvariable](#)
  - [Eindimensionale Arrays](#)
  - [Positionsparameter](#)

- [Spezialparameter](#)
- [Erweiterung](#)
  - [Klammererweiterung](#)
  - [Tildenerweiterung](#)
  - [Parametererweiterung](#)
  - [Kommandosubstitution](#)
  - [Arithmetische Erweiterung](#)
  - [Prozeßsubstitution](#)
  - [Worttrennung](#)
  - [Pfadnamenerweiterung](#)
  - [Quotenreduktion](#)
- [Synonyme](#)
- [Signale](#)
- [Eingabeaufforderung](#)
- [Wenn alles getan ist](#)
- [Eingebaute Shellkommandos](#)
  - [:](#)
  - [alias](#)
  - [bg](#)
  - [bind](#)
  - [break](#)
  - [builtin](#)
  - [bye](#)
  - [cd](#)
  - [command](#)
  - [continue](#)
  - [declare](#)
  - [dirs](#)
  - [disown](#)
  - [echo](#)
  - [enable](#)
  - [eval](#)
  - [exec](#)
  - [exit](#)
  - [export](#)

- [fc](#)
- [fg](#)
- [getopts](#)
- [hash](#)
- [help](#)
- [history](#)
- [jobs](#)
- [kill](#)
- [let](#)
- [local](#)
- [logout](#)
- [popd](#)
- [pushd](#)
- [pwd](#)
- [read](#)
- [readonly](#)
- [return](#)
- [set](#)
- [shift](#)
- [shopt](#)
- [source](#)
- [suspend](#)
- [test](#)
- [time](#)
- [times](#)
- [trap](#)
- [type](#)
- [typeset](#)
- [ulimit](#)
- [umask](#)
- [unalias](#)
- [unset](#)
- [wait](#)
- [Login- und andere Shells](#)
- [Optionen](#)

- [Argumente beim Aufruf der Shell](#)
- [Dateien](#)
- [cat](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Beispiele:](#)
    - [Autor:](#)
- [chgrp](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
    - [Autor:](#)
- [chmod](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Beispiel:](#)
  - [Siehe auch:](#)
    - [Autor:](#)
- [chsh](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
    - [Autor:](#)
- [cksum](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
    - [Autor:](#)
  - [Siehe auch:](#)

- [cmp](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
    - [Autor:](#)
- [comm](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
    - [Autor:](#)
- [compress](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- [cp](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Optionen:](#)
    - [Autor:](#)
- [cpio](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
    - [Autor:](#)
- [csplit](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)

- [Optionen:](#)
- [Beispiel:](#)
- [Siehe auch:](#)
  - [Autor:](#)
- [cut](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Beispiel:](#)
    - [Autor:](#)
- [date](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Die Systemzeit einstellen](#)
  - [Optionen:](#)
  - [Umgebung:](#)
  - [Beispiel:](#)
    - [Autor:](#)
- [dd](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Beispiel:](#)
    - [Autor:](#)
- [df](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
    - [Autor:](#)
- [dirname](#)
  - [Funktion:](#)



- [Syntax:](#)
- [Beschreibung:](#)
- [Siehe auch:](#)
  - [Autor:](#)
- [doshell](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
    - [Autor:](#)
- [du](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
    - [Autor:](#)
- [echo](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
    - [Autor:](#)
- [egrep](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Beispiel:](#)
  - [Siehe auch:](#)
    - [Autor:](#)
- [elvis](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)

- [Überblick](#)
- [Der `visual mode`](#)
  - [`Visual mode` Befehle](#)
  - [Kommandos zum Positionieren des Cursors:](#)
  - [Kommandos zum Ändern von Text:](#)
  - [Arbeiten mit Puffern und Marken:](#)
  - [Kommandos zum Suchen nach Ausdrücken und Wörtern:](#)
  - [Wiederholungs Kommandos:](#)
  - [Sonstige Kommandos:](#)
  - [Eingabemodi](#)
  - [Die Cursortasten in den Eingabemodi](#)
  - [Digraphs](#)
  - [Abkürzungen](#)
  - [Automatisches Einrücken](#)
- [Der `colon mode`](#)
  - [`Colon mode` Befehle](#)
  - [Zeilenangaben](#)
  - [Texteingabe Kommandos](#)
  - [Ausschneiden und Einfügen](#)
  - [Kommandos zum Anzeigen von Text](#)
  - [Kommandos, die auf den gesamten Text wirken](#)
  - [Kommandos zum Editieren einzelner Zeilen](#)
  - [Der undo Befehl](#)
  - [Konfiguration und Status](#)
  - [Kommandos zum Arbeiten mit mehreren Dateien](#)
  - [Zwischen Dateien wechseln](#)
  - [Arbeiten mit einem Compiler](#)
  - [elvis beenden](#)
  - [Datei Ein- und Ausgabekommandos](#)
  - [Verzeichnis Kommandos](#)
  - [Debugging Kommandos](#)
- [Reguläre Ausdrücke](#)
  - [Funktion](#)
  - [Ersetzungen](#)
  - [Optionen](#)

- [Beispiele](#)
- [Die Optionen von elvis](#)
- [Zwischenspeicher \(Puffer\)](#)
  - [Text zwischenspeichern](#)
  - [Einfügen aus einem Puffer](#)
  - [Makros](#)
  - [Wechseln der Arbeitsdatei](#)
- [Autor:](#)
- [elvrec](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
    - [Autor:](#)
- [env](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
    - [Autor:](#)
- [expand](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
    - [Autor:](#)
- [expr](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
    - [Autor:](#)
- [fdformat](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)

- [Autor:](#)

- [file](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)

- [Autor:](#)

- [find](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)
- [Tests:](#)
- [Aktionen:](#)
- [Operatoren:](#)
- [Beispiel:](#)

- [Autor:](#)

- [fold](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)

- [Autor:](#)

- [free](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)

- [Autor:](#)

- [grep](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)
- [Beispiel:](#)

- [Autor:](#)

- [groff](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)
- [Umgebung:](#)
- [Dateien:](#)
- [Beispiel:](#)
- [Siehe auch:](#)

- [Autor:](#)

- [groups](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Siehe auch:](#)

- [gzip](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)
- [Siehe auch:](#)

- [Autor:](#)

- [head](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)

- [Autor:](#)

- [hexdump](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)
- [Siehe auch:](#)

- [hostname](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Siehe auch:](#)
    - [Autor:](#)
- [id](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
    - [Autor:](#)
- [install](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
    - [Autor:](#)
- [join](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
    - [Autor:](#)
- [kill](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
    - [Autor:](#)
- [less](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Kommandos:](#)

- [Optionen:](#)
- [Umgebungsvariable](#)
- [Siehe auch:](#)
  - [Autor:](#)
- [ln](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
    - [Autor:](#)
- [login](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
    - [Autor:](#)
- [logname](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
- [ls](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
    - [Autor:](#)
- [man](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
    - [Autor:](#)
- [mcopy](#)
  - [Funktion:](#)

- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)
- [Siehe auch:](#)
- [mdel](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- [mdir](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- [mformat](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- [mkdir](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
    - [Autor:](#)
- [mkfifo](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
    - [Autor:](#)



- [mmd](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- [more](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
    - [Autor:](#)
- [mrd](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Siehe auch:](#)
- [mread](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- [mt](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Beispiel:](#)
  - [Siehe auch:](#)
    - [Autor:](#)
- [mtools](#)
  - [Funktion:](#)
  - [Beschreibung:](#)
    - [Autor:](#)

- [mv](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
    - [Autor:](#)
- [newgrp](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
    - [Autor:](#)
- [nice](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
    - [Autor:](#)
- [nl](#)
  - [Funktion](#)
  - [Syntax](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
    - [Autor:](#)
- [nohup](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
- [od](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)

- [Autor:](#)
- [passwd](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Siehe auch:](#)
    - [Autor:](#)
- [paste](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
    - [Autor:](#)
- [pr](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
    - [Autor:](#)
- [printenv](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
    - [Autor:](#)
- [ps](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
    - [Autor:](#)
- [pwd](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
- [rm](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)
  - [Autor:](#)
- [rmdir](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
    - [Autor:](#)
- [sed](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
    - [Autor:](#)
- [setfdprm](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
    - [Autor:](#)
- [sleep](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
    - [Autor:](#)
- [sort](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)

- [Autor:](#)
- [split](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
    - [Siehe auch:](#)
    - [Autor:](#)
- [strace](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- [stty](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Beispiel:](#)
    - [Autor:](#)
- [su](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
    - [Autor:](#)
- [sum](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
    - [Siehe auch:](#)
    - [Autor:](#)
- [superformat](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)
- [Siehe auch:](#)
  - [Autor:](#)
- [sync](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
- [tac](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
    - [Autor:](#)
- [tail](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
    - [Autor:](#)
- [tar](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Beispiel:](#)
  - [Siehe auch:](#)
    - [Autor:](#)
- [tee](#)
  - [Funktion:](#)
  - [Syntax:](#)

- [Beschreibung:](#)
- [Optionen:](#)
  - [Autor:](#)
- [touch](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- [tty](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
    - [Autor:](#)
- [uname](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Optionen:](#)
    - [Autor:](#)
- [uniq](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Optionen:](#)
    - [Autor:](#)
- [wall](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
- [wc](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- [who](#)
  - [Funktion:](#)

- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)
  - [Autor:](#)
- [write](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
    - [Autor:](#)
- [Die Kommandos für root](#)
  - [chown](#)
    - [Funktion:](#)
    - [Syntax:](#)
    - [Beschreibung:](#)
    - [Optionen:](#)
    - [Siehe auch:](#)
      - [Autor:](#)
  - [elvprsv](#)
    - [Funktion:](#)
    - [Syntax:](#)
    - [Beschreibung:](#)
      - [Autor:](#)
  - [fdisk](#)
    - [Funktion:](#)
    - [Syntax:](#)
    - [Beschreibung:](#)
    - [Optionen:](#)
    - [Siehe auch:](#)
      - [Autor:](#)
  - [fsck \(Front-End\)](#)
    - [Funktion:](#)
    - [Syntax:](#)
    - [Beschreibung:](#)
    - [Optionen:](#)
    - [Siehe auch:](#)



- [Autor:](#)

- [fsck.ext2 \(e2fsck\)](#)

- [Funktion:](#)

- [Syntax:](#)

- [Beschreibung:](#)

- [Optionen:](#)

- [Siehe auch:](#)

- [Autor:](#)

- [fsck.minix \(fsck\)](#)

- [Funktion:](#)

- [Syntax:](#)

- [Beschreibung:](#)

- [Optionen:](#)

- [Autor:](#)

- [fsck.xiafs \(xfck\)](#)

- [Funktion:](#)

- [Syntax:](#)

- [Beschreibung:](#)

- [Optionen:](#)

- [Siehe auch:](#)

- [Autor:](#)

- [insmod](#)

- [Funktion:](#)

- [Syntax:](#)

- [Beschreibung:](#)

- [Optionen:](#)

- [Siehe auch:](#)

- [Autor:](#)

- [mkboot](#)

- [Funktion:](#)

- [Syntax:](#)

- [Beschreibung:](#)

- [Optionen:](#)

- [Beispiel:](#)

- [Siehe auch:](#)

- [Autor:](#)
- [mkfs \(Front-End\)](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
  - [Autor:](#)
- [mkfs.ext2 \(mke2fs\)](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Autor:](#)
- [mkfs.minix \(mkfs\)](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Autor:](#)
- [mkfs.xiafs \(mkxfs\)](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
  - [Autor:](#)
- [mknod](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Autor:](#)
- [mkswap](#)
  - [Funktion:](#)

- [Syntax:](#)
- [Beschreibung:](#)
- [Beispiel:](#)
- [Optionen:](#)
  - [Autor:](#)
- [mount](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
    - [Autor:](#)
- [rdev](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
    - [Autor:](#)
- [rmmod](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Siehe auch:](#)
    - [Autor:](#)
- [rpm](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Installation neuer Pakete](#)
  - [Upgrade existierender Software](#)
  - [Suche nach Informationen über ein Softwarepaket](#)
  - [Löschen einer installierten Software](#)
  - [Verifizieren der Integrität einer Softwareinstallation](#)
  - [Reparieren einer fehlerhaften Installation](#)
  - [Erzeugen neuer RPM-Pakete](#)
  - [Siehe auch:](#)

- [shutdown](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
    - [Autor:](#)
- [umount](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- [Systemverwaltung](#)
  - [Der privilegierte root-Account](#)
  - [Der Anfang und das Ende](#)
    - [Von Diskette booten](#)
    - [Von Festplatte booten](#)
    - [LILO](#)
      - [Installation](#)
      - [Bootkonzepte 1: wohin mit dem Bootloader?](#)
      - [Bootkonzepte 2: woher mit dem Betriebssystem?](#)
      - [Eine Beispielkonfiguration](#)
      - [Konfigurationsdatei](#)
      - [LILO deinstallieren](#)
    - [Der Bootprompt](#)
    - [Die Vorgänge bei der Kernelinitialisierung](#)
    - [RAM-Disk und Initrd](#)
    - [init](#)
      - [Die Konfigurationsdatei /etc/inittab](#)
      - [Respawn mit Initscript](#)
      - [Die Shellscrip te zur Systeminitialisierung](#)
      - [Auswahl und Änderung des Runlevel](#)
    - [Das System herunterfahren](#)
  - [Laufzeitmodule für den Kernel](#)
    - [Erzeugung und Installation von Kernelmodulen](#)
      - [Zusammenhang zwischen Modulen und Kernelversion](#)

- [Aufbewahrung der Kernelmodule](#)
  - [Laden und Entfernen von Kernelmodulen](#)
    - [Die Konfigurationsdatei `/etc/conf.modules`](#)
    - [Initialisierungsparameter für modulare Gerätetreiber](#)
    - [Entfernen von Kernelmodulen](#)
    - [Automatisches Laden und Entfernen von Modulen durch den Kerneldämon](#)
- [Prozeßordnung](#)
  - [Entstehung der Prozesse: fork und exec](#)
    - [Copy on Write und Demand Loading](#)
  - [Prozeßgruppen, Sessions und kontrollierende Terminals](#)
  - [Prozeßtabelle und Programmumgebung](#)
  - [Abstürzende Programme und hängende Prozesse](#)
    - [Prozesse durch Signale beenden](#)
    - [Zombies und blockierte Prozesse](#)
  - [Systemabsturz](#)
- [Betrachtungen über die Zeit](#)
  - [Linux und Normalzeit](#)
  - [Linux und Zeitzonen](#)
  - [Systemuhr und CMOS-Uhr](#)
  - [Linux und Echtzeit](#)
- [Linux als Mehrbenutzersystem](#)
  - [User, Gruppen und Allgemeinheit](#)
    - [Natürliche Personen und funktionale Rollen](#)
    - [Haupt- und Nebengruppen](#)
  - [Eigentum und Zugriffsrechte](#)
    - [Die Zugriffsrechte als Modus einer Datei](#)
    - [Bedeutung des Zugriffsmodus bei Verzeichnissen](#)
    - [Eigentum an Dateien](#)
    - [Eigentum an Prozessen](#)
    - [Wem gehört was - SUID und SGID Modus bei einem Programm](#)
    - [Besondere Modalitäten für Verzeichnisse](#)
    - [Mandatory Locking](#)
  - [Die Dateiattribute des ext2fs](#)
    - [Schreibzugriff nur zum Anhängen weiterer Daten](#)
    - [Einfrieren einer Datei](#)

- [Geheime Daten gehören in den Reißwolf](#)
- [Gespeichert ist noch nicht gesichert](#)
- [Genauere Betrachtung der Datensicherheit](#)
  - [Authentifizierung](#)
  - [Zusätzliche Sicherheit durch Datenverschlüsselung](#)
    - [Pretty Good Privacy \(PGP\)](#)
    - [DES-Verschlüsselung und Crypto-Filesystem](#)
- [Benutzer eintragen](#)
  - [Eintrag in /etc/passwd](#)
  - [Gruppenzwang](#)
  - [Das Heimatverzeichnis anlegen](#)
- [Partitionen und Dateisysteme](#)
  - [Die Festplatte partitionieren](#)
    - [Hintergrundinformation](#)
    - [Die Partitionstabelle](#)
    - [Benutzung von fdisk](#)
    - [Die Bedeutung des Boot-Flags](#)
    - [Die Bedeutung des Partitionstyps](#)
  - [Das Dateisystem einrichten](#)
  - [Das Dateisystem zusammenbauen](#)
    - [Die Dateisystemtypen von Linux 2.0](#)
    - [Allgemeine Optionen beim Mounten der Dateisysteme](#)
    - [Gemeinsame Optionen für verschiedene Dateisystemtypen](#)
    - [Spezielle Optionen für das EXT2FS](#)
    - [Spezielle Optionen für die FAT-Dateisysteme](#)
    - [Spezielle Optionen für das iso9660 Dateisystem](#)
    - [Spezielle Optionen für Netzwerk-Dateisysteme \(NFS und SMB\)](#)
    - [Feste Vorgaben für die Zusammensetzung des Dateisystems](#)
  - [Die Konsistenz des Dateisystems prüfen](#)
    - [Das Rootfilesystem reparieren](#)
- [Softwaremanagement](#)
  - [Vom Sourcepaket zum fertigen Programm](#)
  - [Paketmanagement mit RPM](#)
    - [Installation und Upgrade einer Software](#)
    - [Untersuchung von RPM-Paketen](#)

- [Direkter Zugriff auf die Daten in der RPM-Datei](#)
- [RPM und die Konfiguration der Software](#)
- [Datensicherung](#)
  - [Medien zur Datensicherung](#)
    - [Magnetbänder im Allgemeinen](#)
    - [Mehrere Dateien \(Archive\) auf einem Magnetband](#)
    - [Dateien \(Archive\) auf mehreren Magnetbändern](#)
    - [Floppystreamer](#)
    - [QIC-Streamer](#)
    - [SCSI-Streamer](#)
    - [Disketten](#)
  - [Methoden der Datensicherung](#)
  - [Backup Software](#)
    - [Datensicherung mit tar](#)
    - [Datensicherung mit afio und cpio](#)
- [Der Druckerdämon lpd](#)
  - [Den lpd erziehen](#)
    - [/etc/printcap](#)
  - [Die Druckerfilter](#)
    - [Der Ausgabefilter](#)
    - [Die Eingabefilter](#)
- [Der Batchdämon crond](#)
  - [Der Terminkalender crontab](#)
    - [Die crontab-Datei](#)
    - [Beispiele:](#)
    - [Umgebungsvariable in der crontab-Datei](#)
    - [Das crontab-Kommando](#)
- [Der Protokollschreiber syslogd](#)
  - [Die Datei /etc/syslog.conf](#)
  - [Beispiele:](#)
- [Recompilieren des Kernels](#)
  - [Entpacken der Quelltexte](#)
  - [Die Kernel-Konfiguration](#)
  - [make xconfig](#)
  - [make menuconfig](#)

- Die Konfigurationsmöglichkeiten im Einzelnen

- Code maturity level options
- Loadable module support
- General setup
- Floppy, IDE and other block devices
- Networking options
- SCSI support
- SCSI low-level drivers
- Network device support
- ISDN subsystem
- CDROM drivers (not for SCSI or IDE/ATAPI drives)
- Filesystems
- Charakter devices
- Sound
- Kernel hacking

- Fremde Welten

- dosemu

- Allgemeines

- Warnung!
- Funktion:
- Syntax:
- Optionen:
- Diskettenlaufwerke
- Das „virtuelle“ Bootlaufwerk
- Festplattenlaufwerke
- Video-Konfiguration
- Tastatur-Konfiguration
- Serielle Schnittstellen
- Terminalunterstützung
- X-Window-Unterstützung

- Verlassen des Emulators

- Wine

- Der iBCS2-Emulator

- Wie Sie iBCS2 bekommen und installieren können
- Shared Libraries



- [SVR3/COFF](#)
- [SVR4/ELF](#)
- [Gerätedateien](#)
- [Programme installieren](#)
- [Datenreisen und reisende Daten](#)
  - [Technische Voraussetzungen](#)
    - [Modems](#)
      - [Die technischen Daten](#)
      - [Kabelsalat](#)
    - [Serielle Schnittstellen](#)
      - [16550 UART](#)
      - [Mehrportkarten und Karten auf anderen Adressen/Interrupts](#)
  - [Kontaktaufnahme](#)
    - [Die `minicom` Terminalemulation](#)
      - [Installation und Konfiguration](#)
      - [Anwählen eines anderen Rechners](#)
      - [Automatische Frage- und Antwort-Spiele](#)
      - [Dateien übertragen](#)
    - [term](#)
      - [Kurzer Überblick](#)
      - [Der `term` Server](#)
      - [Leitungstransparenz](#)
      - [trsh](#)
      - [tupload](#)
      - [tredir](#)
      - [txconn](#)
      - [tmon](#)
  - [Seriell einloggen](#)
    - [Das richtige `getty`](#)
      - [Device-Handling](#)
      - [Lockfiles](#)
    - [getty\\_ps](#)
      - [Syntax:](#)
      - [Optionen:](#)
      - [Defaults-Dateien](#)

- [gettydefs](#)
- [UUCP - Das Internet der Armen Leute](#)
  - [UUCP-Anschluß - aber wie?](#)
  - [Was kann UUCP?](#)
  - [Wie sicher ist UUCP?](#)
  - [Taylor-UUCP](#)
  - [Überblick über die Konfigurationsdateien](#)
  - [Log-Dateien](#)
  - [Die config-Datei](#)
  - [Die sys-Datei](#)
    - [Telefonnummer und Login-Information](#)
    - [Erlaubte Zeiten](#)
    - [Das Chat-Skript](#)
    - [Protokoll-Parameter](#)
  - [Die port-Datei](#)
  - [Die dial-Datei](#)
  - [Testen der Konfiguration](#)
  - [Regelmäßige Verbindungen](#)
- [Elektronische Post mit \*smail\*](#)
  - [Wie sieht eine Mail denn nun aus?](#)
  - [Adressen, Adressen, Adressen](#)
  - [Taler, Taler, Du mußt wandern](#)
  - [Email-Software unter Linux](#)
  - [Installation von \*smail\*](#)
    - [Die config-Datei](#)
  - [Elektronische Post mit \*elm\*](#)
    - [\*elm\*-Konfiguration](#)
  - [Ein Test](#)
- [Usenet News](#)
  - [Die technischen Details](#)
  - [News-Software](#)
- [INN](#)
  - [INN und IP-Networking](#)
  - [Administrativer Kleinkram](#)
  - [Globale Parameter](#)

- [Die Dateien \*active\* und \*newsgroups\*](#)
  - [Wer bekommt was - \*newsfeeds\*](#)
  - [Leben und Sterben des \*innd\*](#)
  - [Ein ausgehender Newsfeed über UUCP](#)
  - [Löschen von Artikeln mit \*expire\*](#)
  - [Control-Messages](#)
  - [Overview-Dateien](#)
  - [Und jetzt ohne Hände...](#)
  - [Der Newsreader \*tin\*](#)
  - [Das Point-to-Point-Protokoll \(PPP\)](#)
    - [Voraussetzungen](#)
      - [Die andere Seite](#)
      - [Die treibende Kraft im Kern](#)
      - [Der gute Geist](#)
      - [Das Vorspiel](#)
    - [Die Konfiguration des \*pppd\*](#)
      - [Die options-Datei](#)
      - [Authentifizierung via PAP oder CHAP](#)
    - [Die Verbindung starten](#)
    - [Die Verbindung beenden](#)
- [Die Installation von Linux](#)
  - [Planung](#)
    - [Eine eigene Partition für Linux finden](#)
    - [Wieviel Platz braucht Linux?](#)
    - [Linux auf mehreren Partitionen](#)
    - [Was Sie noch brauchen](#)
  - [Durchführung](#)
    - [Die Installation im Überblick](#)
      - [Herstellung einer Bootdiskette](#)
      - [Booten](#)
      - [Die Festplatte partitionieren](#)
      - [Swappartition und Dateisysteme einrichten](#)
      - [Das Kopieren der Daten](#)
      - [Konfiguration und Erzeugung der Bootdiskette](#)
- [Dateisysteme](#)

- [Das Minix-Dateisystem](#)
  - [I-Nodes](#)
  - [Verzeichnisse](#)
  - [Superblock und Bitmaps](#)
    - [Datenzonen und Plattenblöcke](#)
  - [Links](#)
    - [Links auf Verzeichnisse](#)
- [Die neuen Dateisysteme](#)
  - [Die Zeitmarken in der I-Node](#)
  - [Entwicklung](#)
  - [Das zweite erweiterte Dateisystem \(ext2fs\)](#)
    - [Das Valid-Flag](#)
    - [Dateiattribute](#)
    - [Gruppierung der Datenzonen](#)
  - [Das xiafs](#)
- [Bewertung](#)
- [Spezialdateisysteme](#)
  - [Das Prozeßdateisystem](#)
    - [Die Prozeßverzeichnisse](#)
  - [Das ISO-9660-Dateisystem](#)
  - [umsdos](#)
- [Die Linux-Console](#)
  - [Der Bildschirm](#)
    - [Einen neuen Zeichensatz laden](#)
      - [Reset der Console](#)
  - [Die Tastatur](#)
    - [Die Tastaturtabelle](#)
      - [Die Belegung der keycodes](#)
      - [Die Definition der Funktionstasten](#)
      - [Die zusammengesetzten Zeichen \(Diacriticals\)](#)
    - [Metazeichen](#)
- [Locales und Native Language Support](#)
  - [Überblick](#)
  - [Anwendung von Locales](#)
  - [Erzeugung und Installation der Regelsätze](#)

- [Native Language Support \(NLS\)](#)
- [Literaturliste](#)
  - [Linux](#)
  - [Unix allgemein](#)
  - [Unix intern](#)
  - [Unix Systemverwaltung](#)
  - [X Window System](#)
  - [Vernetzung](#)
  - [Programmierung](#)
- [Was ist eigentlich die GPL](#)
  - [Das Wesentliche in Kürze](#)
  - [Die GPL im Einzelnen](#)
- [Bestellung des Linux Anwenderhandbuches](#)
- [Index](#)

# Grundlagen

---

- [Geschichte](#)
  - [Steinzeit](#)
  - [Unix](#)
  - [Minix](#)
  - [Linux](#)
- [Das Betriebssystem](#)
- [Die ersten Schritte](#)
  - [Verzeichnisse und Dateien](#)
  - [Datenströme](#)
  - [Eingabehilfen von der Shell](#)
  - [Virtuelle Terminals und Hintergrundprozesse](#)
- [Zahlensysteme](#)

## Subsections

- [Steinzeit](#)
  - [Unix](#)
  - [Minix](#)
  - [Linux](#)
- 

# Geschichte

## Steinzeit

In grauer Vorzeit wurden Elektronenrechner mit Lochkarten gefüttert. Die Programme und die Daten wurden in die Karten gestanzt und als Pakete in den Kartenleser geschoben, die Ergebnisse auf dem Drucker ausgegeben. Klar, daß solche Rechner nur ein einziges Programm für einen einzigen Anwender bearbeiten konnten. Das Betriebssystem, wenn man es überhaupt so nennen kann, bestand aus den fest verdrahteten Funktionen des Rechenwerkes.

Auch als die Daten und Programme von Magnetspeichern gelesen und die Ergebnisse auf Bildschirmen statt auf Druckern ausgegeben wurden, blieb es bei der Batchverarbeitung durch den Großrechner. Zwar konnten bereits mehrere Jobs in einem Stapel (Batch) geladen werden, es blieb aber bei der sequentiellen Bearbeitung der Jobs.


Durch die wachsende Rechenleistung und vor allem durch die bessere Ausstattung mit Arbeitsspeicher können seit der dritten Rechnergeneration mehrere Jobs gleichzeitig in den Arbeitsspeicher geladen werden. Auf diese Weise kann die wertvolle Rechenzeit (CPU-Zeit) auch dann ausgenutzt werden, wenn ein Job beispielsweise auf neue Daten vom Magnetband warten muß (indem das Betriebssystem einen anderen Job im Speicher laufen läßt, bis die Daten vom Magnetband geladen sind).

Die Bearbeitung kompletter Jobs wurde in den sechziger Jahren durch das quasiparallele Bearbeiten mehrerer Programme im Timesharing-Verfahren abgelöst. Dabei werden mehrere Programme geladen und allen Programmen wird nach einem festgelegten Verfahren Rechenzeit in Zeitscheiben zugeteilt. Mit dieser auch als *Multitasking* bezeichneten Methode können interaktive Programme zur Datenerfassung gleichzeitig mit den rechenzeitintensiven Programmen zur Datenauswertung laufen. Durch diese Mischung wurde eine weitaus bessere Auslastung der teuren Hardware möglich.

Für die interaktive Arbeit am Großrechner werden Bildschirm-/Tastatur-Kombinationen, sogenannte Terminals, angeschlossen. Ein Großrechner, der gleichzeitig mehrere solcher Terminals bedienen kann, wird als *Multiusersystem* bezeichnet. Die Terminals verarbeiten die Daten nicht selber, sondern sie schicken die auf der Tastatur eingegebenen Zeichen unverändert an den Zentralrechner und stellen die von diesem Rechner gesendeten Zeichen auf dem Bildschirm dar. Die Terminals sind also "dumme" Arbeitsstationen. Moderne Terminals, zum Beispiel das vt100 Terminal von DEC, erkennen allerdings neben den normalen Buchstaben zusätzlich verschiedene Sonderzeichen zum


Löschen eines Buchstabens oder des gesamten Bildschirms, zum Positionieren der Einfügemarke, für invertierte Darstellung und so weiter.

# Unix

Das erste Unix  wurde 1969 von Ken Thompson und Dennis Ritchie bei den Bell Laboratories (AT&T und Western Electric) geschrieben (um auf einer wenig benutzten DEC PDP-7 in einer Ecke Space Travel zu spielen ...). Bis 1971 war es als Version 1 auf eine PDP-11 portiert; Version 4 wurde 1973 von dem ursprünglich in Assembler geschriebenen Quelltext fast vollständig in die eigens dafür entwickelte Hochsprache C umgeschrieben.


Weil AT&T durch Verträge mit der US-Bundesregierung (per Gerichtsbeschluß?) daran gehindert war, Unix zu vermarkten, gab sie für Lehr- und Forschungszwecke Sourcelizenzen zu sehr günstigen Konditionen (ein paar hundert Dollar) an Universitäten weiter. Dieser möglicherweise gut gemeinte, jedenfalls außerordentlich geschickte Schritt führte zu einer sehr dynamischen Entwicklung von Unix. Es verbreitete sich schnell, und aus den Universitäten flossen viele Ideen in die Entwicklung des Systems ein.

Seit Version 6 (1975) hat AT&T aber auch kommerzielle Lizenzen für Unix verkauft. Spätestens seit dem Jahr 1984, als ein weiteres Gerichtsurteil AT&T die Vermarktung von Software erlaubte, wurde Unix als System V unter rein kommerziellen Gesichtspunkten verbreitet.

Gerade wegen seiner Verbreitung an den Universitäten hat es sich im kommerziellen Sektor schnell durchgesetzt.  Da fast komplett in C geschrieben, ist es auf praktisch alle Großrechnerarchitekturen portiert. Für Softwareanbieter ist Unix ein (zur Hardwareseite) offenes System. Aus eifersüchtiger Sorge um die Quelltexte hat AT&T die tiefgreifendere Öffnung auf Systemebene abgeblockt.

# Minix

Im Jahr 1987 hat Andrew Tanenbaum, Professor an der Freien Universität von Amsterdam, ein Lehrbetriebssystem für PC veröffentlicht, das ohne jeden AT&T Code die Funktionalität von Unix Version 7 hat und als Quelltext für wenig Geld zu kaufen ist.

Minix ist keine echte Basis für Anwenderprogramme; aber es ist ein sehr lehrreiches Spielzeug. Ein Lebensnerv von Minix ist das USENET, wo in der Gruppe comp.os.minix alle Neuigkeiten, Fragen und Antworten zu Minix ausgetauscht werden. Hier werden auch Veränderungen am Betriebssystem (dem **Kernel**)  veröffentlicht und gelegentlich ganze Programme verschickt. Hier tummelt sich eine weltweit aktive Minix-Gemeinde und entwickelt das Betriebssystem und die Anwendungen drumherum.

Minix wurde auf den Atari ST, den Amiga und den Apple Macintosh portiert; und es wurde an die erweiterten Möglichkeiten des Intel-386-Prozessors angepaßt. Auf diese Weise konnte die lästige Beschränkung auf 64 Kilobytes Speicher je Prozeß überwunden werden.

Es sollten und sollen aber andere Beschränkungen für Minix bestehen bleiben, auf denen Andrew Tanenbaum als Autor besteht. So ist Minix preiswert, aber nicht frei kopierbar. Und es werden keine grundlegenden Veränderungen am Kernel zugunsten von Anwendungen vorgenommen, was zum Beispiel die Portierung der unter Unix weit verbreiteten grafischen Benutzeroberfläche, des X




Window Systems, ausschließt.

Diese Einschränkungen sind im Sinne eines Lehrbetriebssystems sinnvoll, das Bedürfnis nach einem vollwertigen und freien Betriebssystem für die modernen 386-PC's bleibt aber unerfüllt.

# Linux

Im März 1991 fing Linus Benedict Torvalds in Helsinki damit an, die Möglichkeiten des Intel-386-Prozessors in seinem neuen PC zu studieren. Er hatte das 386er Minix installiert - und damit das C-Entwicklungssystem der Free Software Foundation.

Nur ein halbes Jahr später war aus den Assemblerstudien ein kleines, lauffähiges Betriebssystem entstanden. Als Linus im September 1991 die erste Version (0.01) von Linux an interessierte Minixler verschickte, mußte es noch unter Minix übersetzt und installiert werden. Diese Verbindung von Minix und Linux hat sich aber nicht im Quelltext niedergeschlagen. 

Linus Torvalds hat seine eigene Entwicklung von Anfang an frei angeboten. Jeder kann die Quelltexte bekommen und daran mitarbeiten.

Die im Januar 1992 herausgegebene Version 0.12 war bereits ein stabil laufender Kernel. Es gab den GNU C-Compiler, die bash, uemacs und viele der GNU Utilities. Dieses Betriebssystem wurde in comp.os.minix angekündigt und per anonymous FTP weltweit verteilt.

Die Zahl der Programmierer, Tester und Unterstützer wuchs in diesen Tagen so schnell, daß die Kommunikation per eMail nicht mehr ausreichte und nach dem Beispiel von comp.os.minix die Rubrik alt.os.linux im USENET eingerichtet wurde. Dieses Medium und der anonyme FTP Service im Internet ermöglichten eine Programmentwicklung, wie sie in den Vorstandsetagen der mächtigen Softwareschmieden erträumt wird. Innerhalb weniger Monate wurde aus dem weitgehend in Assembler geschriebenen Minikernel ein ausgewachsenes Betriebssystem mit vollständiger Unix Funktionalität.

Die weitgehende POSIX-Konformität von Linux und die umfangreichen C-Bibliotheken des GNU-C-Compilers erleichtern die Portierung von Unix- oder BSD-Software auf ein Maß, das diesen Namen eigentlich nicht verdient. Praktisch das gesamte Angebot an freier Software läuft auch unter Linux. Dank der enormen Leistungen, die bereits seit vielen Jahren von der Free Software Foundation, dem X Consortium, den Universitäten und ungezählten Organisationen und Einzelpersonen in diesem Bereich erbracht wurden, haben die Linux-Distributionen heute einen Umfang erreicht, der die Angebote kommerzieller Unixe leicht in den Schatten stellt.

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [Das Betriebssystem](#) **Up:** [Grundlagen](#) **Previous:** [Grundlagen](#)


*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

# Das Betriebssystem

Jeder Computer besteht aus mehreren Komponenten und Geräten, der sogenannten Hardware. Unter anderem sind das die Zentrale-Prozessor-Einheit (CPU), der Arbeitsspeicher, die Festplatte und die Diskettenlaufwerke, der Bildschirm und die Tastatur, der Drucker und das Modem. Die Komponenten sind nur lose miteinander verknüpft. Philosophisch betrachtet, stellen sie eine universelle Maschine dar, die erst durch ein konkretes Anwenderprogramm zu einer simulierten Schreibmaschine, einer Lohnbuchhaltung, einem Schachspiel oder einer simulierten Mondlandefähre wird.

Um den Programmierer eines Anwenderprogramms von den Einzelheiten der Hardwareprogrammierung zu entlasten, werden die Komponenten durch das sogenannte Betriebssystem verwaltet. Im Idealfall stellt das Betriebssystem alle Dienste der Hardware in einer abstrakteren Form zur Verfügung, ist also eine Art Hardwareerweiterung. Im Multiuser/Multitasking-System hat das Betriebssystem zusätzlich die Aufgabe, konkurrierende Hardwarezugriffe verschiedener Benutzer oder Prozesse zu verwalten.

Das Betriebssystem muß, wie jedes andere Programm auch, zur Laufzeit im Arbeitsspeicher geladen sein. Bei einigen Computern steht das Betriebssystem in dauerhaften Speicherbausteinen, sogenannten ROM's. Die IBM-kompatiblen PC's laden den größten Teil des Betriebssystems von Diskette oder

Festplatte.  Das Betriebssystem ist das erste Programm, das nach dem Einschalten des Rechners automatisch geladen und gestartet wird. Wenn es einmal geladen ist, bleibt das Betriebssystem im Arbeitsspeicher, bis der Rechner ausgeschaltet wird.

Gemeinsam mit dem Betriebssystem stehen normalerweise noch eine Reihe von Programmen zur Verfügung, mit denen Sie übergeordnete und auf allen Systemen notwendige Aufgaben ausführen können, wie beispielsweise eine Diskette formatieren oder eine Datei ausdrucken. Ein besonders wichtiges Programm des in diesem Sinne erweiterten Betriebssystems ist der Kommandozeileninterpreter, sozusagen die erste Benutzeroberfläche des Systems. Der Kommandozeileninterpreter liest Ihre Befehle von der Tastatur und führt die darin formulierten Befehle aus. Erst dieses Systemprogramm ermöglicht den universellen Einsatz des Computers, der damit jedes installierte Programm laden und ausführen kann. Bei MS-DOS wird diese Aufgabe von dem allseits bekannten Programm COMMAND.COM erfüllt. Unter Linux gibt es nicht nur ein einziges Programm zu diesem Zweck, vielmehr gibt es eine ganze Klasse, die sogenannten *Shells*. Die Shells erfüllen alle den gleichen Zweck, einige bieten Ihnen zusätzlich eine Vielzahl von komfortablen Bedienhilfen.

Zur optimalen Ausnutzung der teuren Hardware wurden für Großrechner schon sehr früh Betriebssysteme entwickelt, die mehreren Anwendern gleichzeitig die Systembenutzung ermöglichen. Diese als *Multiuser/Multitasking* bezeichnete Eigenschaft war für das Betriebssystem der ersten PC's (1981) überflüssig. Deren Vorteil bestand in dem niedrigen Preis, der es ermöglichte, jeden Arbeitsplatz mit einem eigenen Rechner auszustatten. Dieser Preisvorteil wurde mit einem vergleichsweise niedrigen Leistungsniveau erkauft. Die Prozessoren der ersten PC-Generation waren weder von ihrer Architektur noch von ihrer Performance her in der Lage, mehr als eine Aufgabe zur gleichen Zeit zu erfüllen. Ein moderner 386er Rechner wird dagegen von einem normalen Anwenderprogramm nicht ausgelastet. Die meiste Zeit verbringt das Betriebssystem damit, auf den



nächsten Tastendruck des Benutzers zu warten. Aus diesem Grund werden jetzt auch Multitasking-Betriebssysteme für PC angeboten.


Auch wenn der PC weiterhin ein Arbeitsplatzrechner bleibt, also zu jeder Zeit von nur einem natürlichen Benutzer gebraucht wird, hat die gleichzeitige Bearbeitung mehrerer Programme große Vorteile: Zeitaufwendige automatische Prozesse oder Dienste, wie beispielsweise das Übersetzen von Programmen oder die Übertragung von Daten per Modem, können "im Hintergrund" ablaufen und im Vordergrund gleichzeitig interaktive Arbeiten fortgesetzt werden.

Die Anforderungen an ein Mehrbenutzerbetriebssystem unterscheiden sich grundlegend von einem Einbenutzersystem:

- Es müssen konkurrierende Hardwarezugriffe verhindert beziehungsweise verwaltet werden.
- Es müssen die privaten Daten der Benutzer geschützt werden.
- Die Speicherbereiche der Anwenderprogramme müssen vor ungewollten Veränderungen durch andere Programme geschützt werden.

Als echtes Mehrbenutzerbetriebssystem verwaltet Linux die Systemkomponenten sehr restriktiv. Es erlaubt den Anwenderprogrammen prinzipiell keinen direkten Zugriff auf die Hardware.

Um diese Einschränkung durchzusetzen, werden die Anwenderprogramme durch das Betriebssystem (den Kernel) kontrolliert. Wenn ein Programm vom Benutzer aufgerufen wird, lädt der Kernel die ausführbare Datei  in den Arbeitsspeicher und macht daraus einen Prozeß. Dieser Prozeß erhält bei seiner Entstehung einen logischen Adreßraum , in dem zuerst der Programmtext und die initialisierten Daten des Programms dargestellt werden, in dem das Programm aber auch seine variablen Daten ablegen kann. Die logischen Adressen werden vom Betriebssystem auf die physikalischen Adressen des Arbeitsspeichers abgebildet. Wenn das Programm auf eine (logische) Speicheradresse zugreift, muß diese Adresse erst in die physikalische Adresse umgewandelt werden. Dadurch kann das Betriebssystem unberechtigte Zugriffe auf den Adreßraum anderer Prozesse oder auf die Hardwarekomponenten feststellen und unterbinden (durch das Signal *SIGSEGV*) .

Die einzige Möglichkeit, auf die Systembereiche außerhalb des eigenen Adreßraums zuzugreifen, bietet der Kernel den Programmen auf Benutzerebene durch die sogenannten Systemaufrufe (system calls) . Linux bietet ca. 150 solcher Systemaufrufe an.  Dieses von Unix übernommene Prinzip erscheint auf den ersten Blick vielleicht als Hindernis und Einengung. Bei genauerer Betrachtung stellt man aber die enormen Vorteile fest. So ist es der Übereinstimmung mit den Systemaufrufen des Unix System V zu verdanken, daß praktisch alle Unix-Programme sofort unter Linux übersetzt werden können. Die unterschiedlichen Hardwarevoraussetzungen aller unterschiedlichen Systeme werden allein vom Kernel aufgefangen.

Wie bereits gesagt, arbeitet jedes Anwenderprogramm in einem logischen Speichersegment. Dieses Speichersegment ist in Speicherseiten zu je 4 Kilobyte unterteilt und wird vom Betriebssystem seitenweise auf den physikalischen Arbeitsspeicher abgebildet (mapping). Wenn mehr Programme gestartet werden, als auf einmal in den Arbeitsspeicher passen, kann der Kernel einzelne Speicherseiten aus dem physikalischen Adreßraum auf Festplatte auslagern (swapping) . Wenn das dadurch unvollständige Programm wieder auf eine Adresse der ausgelagerten Speicherseite zugreifen will, wird sie automatisch zurückgeladen. Dank der MMU (Memory Management Unit) des 386-Prozessors werden die grundlegenden Funktionen dieser aufwendigen Speicherverwaltung bereits durch die CPU erledigt. Die intensive Ausnutzung spezieller Prozesseigenschaften macht Linux zu einem außerordentlich schnellen Betriebssystem.

---




## Subsections

- [Verzeichnisse und Dateien](#)
    - [Das aktuelle Verzeichnis](#)
    - [Namen und Pfade](#)
    - [Das Verzeichnis wechseln](#)
    - [Den Inhalt eines Verzeichnisses anzeigen](#)
    - [Eigentum und Zugriffsrechte](#)
    - [Verzeichnisse erstellen und löschen](#)
    - [Dateien erstellen, verschieben und löschen](#)
    - [Dateien anzeigen](#)
  - [Datenströme](#)
    - [Verknüpfung mehrerer Kommandos: Pipelines](#)
  - [Eingabehilfen von der Shell](#)
  - [Virtuelle Terminals und Hintergrundprozesse](#)
- 

# Die ersten Schritte

Anders als bei einem Einbenutzerbetriebssystem wie DOS können Sie bei Linux nicht einfach den Rechner einschalten und drauflosarbeiten. Die Vorteile der Mehrbenutzerfähigkeit werden durch eine etwas kompliziertere Bedienung erkaufte: als Benutzer oder ``User" brauchen Sie einen Benutzerbereich oder ``Account" mit einem Usernamen und einem Paßwort. Mit dem Usernamen melden Sie sich jedesmal, wenn Sie beginnen mit Ihrem Linux-Rechner zu arbeiten, bei dem

Betriebssystem an. Dieser Vorgang wird auch als ``Einloggen" oder ``Login" bezeichnet.  Diese Anmeldung ist notwendig, damit eine eindeutige Zuordnung von Dateien zu ihren Eigentümern hergestellt werden kann und um den unberechtigten Zugriff auf Dateien anderer Eigentümer zu verhindern. Nach der Meldung


login: \_

müssen Sie Ihren Benutzernamen eingeben und mit der RETURN Taste abschließen. Die daraufhin erscheinende Frage nach dem

Password:

muß korrekt mit dem nur Ihnen bekannten Paßwort beantwortet werden. Bei der Eingabe des Paßwortes erscheinen die Zeichen nicht auf dem Bildschirm. Wenn das System die Anmeldung akzeptiert, wird eine sogenannte *Session* eröffnet. Das bedeutet für Sie als interaktiver Benutzer, daß der Kommandozeileninterpreter (die Shell) gestartet wird, so daß Sie über die Tastatur Kommandos eingeben und Programme aufrufen können.

Wenn die Shell bereit ist, Kommandos von der Tastatur zu lesen, gibt sie eine charakteristische Zeichenkette als Eingabeaufforderung aus, den sogenannten ``*Prompt*``. Ähnlich wie das von DOS bekannte `C:\>` kann sich der Shellprompt unter Linux je nach Einstellung und aktueller Systemumgebung verändern. Beispielsweise kann der Rechnername, Ihr Benutzername oder das aktuelle Arbeitsverzeichnis angezeigt werden. Was genau der Prompt anzeigt, hängt von der Shell ab, die Sie benutzen und von der individuellen Einstellung, die Sie oder Ihre Systemverwalterin vorgenommen hat. Wie Sie die Prompts der bei den meisten Linux-Systemen als Standardshell installierten Bourne Again Shell (`bash`) verändern können ist [hier](#) beschrieben.

Die Kommandos werden zeilenweise von der Tastatur gelesen und anschließend von der Shell interpretiert, die die entsprechenden Befehle ausführt. Jede Kommandozeile wird durch die `RETURN` Taste abgeschlossen. Jedes Kommando besteht aus einem oder mehr Wörtern, die durch Leerzeichen voneinander getrennt werden. Das erste Wort ist immer der Kommandoname. Der Kommandoname entspricht exakt dem Namen der ausführbaren Datei.  Eine genauere, allerdings auch formalistischere Erklärung der Kommandozeile finden Sie in der Einleitung zu den [Kommandobeschreibungen](#) und in der genauen Beschreibung der Standardshell [bash](#).

## Verzeichnisse und Dateien

Die im Arbeitsspeicher (RAM) des Rechners gehaltenen Daten, seien es Programme, Texte, Bilder oder Zahlen, gehen alle mit dem Ausschalten der Stromversorgung verloren. Aus diesem Grund werden alle wichtigen Daten auf einem dauerhaften Speichermedium gehalten. Dieses Medium ist normalerweise die Festplatte, es kann aber auch ein Diskettenlaufwerk oder eine CD-ROM sein.

Um die Vielzahl der anfallenden Daten rationell und übersichtlich speichern zu können, bietet Linux, wie alle modernen Betriebssysteme, die Möglichkeit, die auf Festplatten, Disketten usw. gesicherten Daten in Verzeichnissen zu organisieren.

### Das aktuelle Verzeichnis

Als Benutzer eines interaktiven Computers „befinden“ Sie sich ständig in einem Verzeichnis. Dieses Verzeichnis, das **aktuelle Verzeichnis**, oder Arbeitsverzeichnis ist der relative Bezugspunkt für all Ihre Aktionen. Jedes Kommando, das Sie aufrufen, ist mit diesem Verzeichnis verbunden. Wenn Sie nicht ausdrücklich etwas anderes bestimmen, werden Dateien zum Öffnen in diesem Verzeichnis gesucht oder neue Dateien hier angelegt.

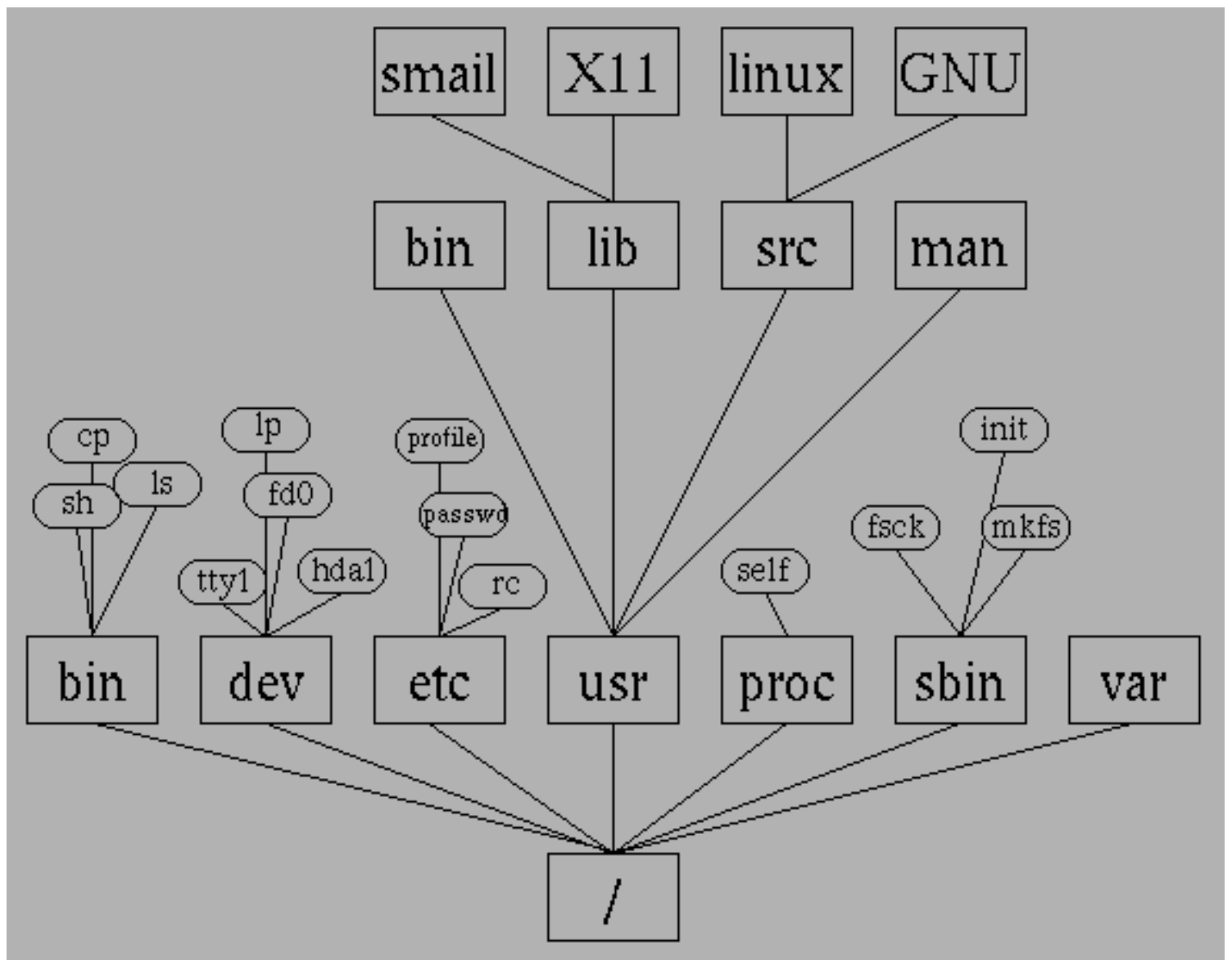
Jedes Verzeichnis kann Dateien und weitere Verzeichnisse enthalten. Der Name des aktuellen Verzeichnisses wird mit dem Kommando `pwd` (`print working directory`) angezeigt:

```
$ pwd
/usr/src/linux
$ _
```

Das aktuelle Verzeichnis ist eines unter vielen Verzeichnissen im System. Alle Verzeichnisse und die darin enthaltenen Dateien zusammen werden als Dateisystem bezeichnet. Das Dateisystem ist mit einem Baum zu vergleichen, dessen Äste und Zweige die Verzeichnisse sind und dessen Blätter die Dateien (siehe Abbildung 1.1). Wenn Sie sich vorstellen, daß jeder Ast, jeder Zweig und jedes Blatt einen eigenen Namen haben, können Sie schnell und leicht nachvollziehbar jeden einzelnen Ort in dem



Baum benennen, indem Sie den Weg oder *„Pfad“* von der Wurzel des Baumes ab beschreiben. Wenn Sie so einen Pfad aufschreiben, werden die einzelnen Namen für Verzeichnisse und Dateien, durch einen einfachen Schrägstrich *„/“* (Slash) voneinander getrennt. Das Wurzelverzeichnis oder Rootdirectory wird durch einen einzelnen Slash *„/“* bezeichnet.



**Abbildung:** Dateibaum (Ausschnitt)

Anders als bei DOS, wo das Dateisystem in Laufwerke unterteilt ist, auf denen jeweils eigene Verzeichnisbäume existieren können, arbeiten Sie unter Linux mit einem einzigen Verzeichnisbaum. Sie brauchen sich als Benutzer nicht darum zu kümmern, wo sich eine bestimmte Datei physikalisch befindet, das Betriebssystem stellt alle verfügbaren Dateien und Verzeichnisse in einem einzigen logischen Baum zusammen.

## Namen und Pfade

Jede Datei hat einen **Dateinamen**, mit dem sie in (mindestens) einem Verzeichnis eingetragen ist. Natürlich kann jeder Dateiname in einem Verzeichnis nur einmal vergeben werden. Allerdings kann der gleiche Name in einem anderen Verzeichnis für eine andere Datei stehen. Wenn Sie mit einer Datei arbeiten wollen, die in einem anderen als dem aktuellen Verzeichnis liegt, reicht es deshalb nicht, einfach den Dateinamen anzugeben. Stattdessen muß die Datei mit ihrem eindeutigen **Pfadnamen** angesprochen werden.

Dieser Pfadname enthält neben dem eigentlichen Dateinamen noch eine ``Wegbeschreibung'', in der durch ein Reihe von **Verzeichnisnamen** der genaue Ort der Datei im Verzeichnisbaum angegeben wird. Die Reihe der Verzeichnisnamen wird als Pfad bezeichnet. Die einzelnen Verzeichnisnamen werden alle durch `/ (Slash) Zeichen getrennt.

Um eine Datei in einem anderen Verzeichnis oder auch einfach nur das andere Verzeichnis anzusprechen, kann der Pfadname entweder relativ zum aktuellen Verzeichnis oder als **absoluter** (Pfad-) Name vom Wurzelverzeichnis aus angegeben werden. Der absolute Pfad beginnt immer mit einem Slash, dem ``Namen" des Wurzelverzeichnisses.

Der **relative** Pfad beginnt im aktuellen Verzeichnis und gibt von dort aus den Weg zum Ziel im Verzeichnisbaum an. Weil jedes Kommando automatisch das aktuelle Verzeichnis als Bezugspunkt für seine Aktionen nimmt, braucht das Arbeitsverzeichnis selbst meistens nicht ausdrücklich angegeben zu werden. Für die Fälle, in denen das trotzdem nötig ist, hat jedes Verzeichnis einen Eintrag `.', der das Verzeichnis selbst bezeichnet.

Ein zweiter Eintrag, `..', der auch in jedem Verzeichnis zu finden ist, bezeichnet das nächste Verzeichnis ``zurück" in Richtung Wurzel. Um bei einem relativen Pfadnamen im Verzeichnisbaum in Richtung Wurzel zu gehen, muß anstelle eines regulären Verzeichnisnamens dieser ``Link" zum nächsttieferen Verzeichnis `..' angegeben werden.

Neben dem Bild des Verzeichnisbaums gibt es noch das einer Hierarchie. Damit wird im Prinzip die gleiche Ordnung erzeugt, aber in genau umgekehrter Richtung. Wenn ein Verzeichnis beim Baum näher an der Wurzel, also tiefer ist, wird es in der Hierarchie höher eingeordnet. In diesem Buch verwenden wir in der Regel das Bild des Baumes und benutzen die damit verbundenen räumlichen Ordnungsrelationen.

## Das Verzeichnis wechseln

Sie können das aktuelle Verzeichnis mit dem `cd` (change directory) Kommando wechseln. Zum Beispiel bringt Sie das Kommando

```
$ cd /  
$ _
```

in das Wurzelverzeichnis. Das Kommando

```
$ cd /dev  
$ _
```

wechselt in das Verzeichnis dev. Der Schrägstrich (Slash) bezeichnet dev vom Wurzelverzeichnis aus. Es können auch mehrere Verzeichnisnamen zu einem Pfadnamen verkettet werden. Um die einzelnen Namen voneinander zu trennen, werden weitere Slashes verwendet. Das Kommando

```
$ cd /usr/lib/emacs  
$ _
```



bringt Sie in das Verzeichnis **emacs**. Um von dort aus in das Verzeichnis **lib** zu gelangen, können Sie entweder wieder den *absoluten* Pfad vom Wurzelverzeichnis aus angeben

```
$ cd /usr/lib
$ _
```

oder Sie benutzen den *relativen* Pfad vom aktuellen Verzeichnis aus:

```
$ cd ../..
$ _
```

Bei diesem Beispiel wird das aktuelle Verzeichnis `..` ausdrücklich angegeben. Das ist für das Wechseln des Arbeitsverzeichnisses eigentlich nicht notwendig, die folgende Variante des Beispiels erzielt das gleiche Ergebnis:


```
$ cd ..
$ _
```


Wenn Sie häufig in bestimmte Verzeichnisse wechseln müssen, können Sie dem in der `bash` enthaltenen `cd`-Kommando in der Umgebungsvariablen `CDPATH` eine Liste von absoluten Verzeichnisnamen angeben, in deren Unterverzeichnisse Sie dann durch Angabe des einfachen Verzeichnisnamen wechseln können.

## Den Inhalt eines Verzeichnisses anzeigen

Um den Inhalt eines Verzeichnisses anzuzeigen, gibt es das Kommando `ls` (list) .

```
$ ls /bin
arch      compress  echo      gzip      ls         ps        su
bash      cp        ed        hostname  mkdir     pwd       sync
cat       date     false     kill      mknod     rm        tar
chgrp     dd       free     killall   mv        rmdir     true
chmod     df       ftp      ln        netstat   sh        uname
chown     du       gunzip   login     ping      stty      zcat
$ _
```

zeigt beispielsweise alle Dateien des **bin** Verzeichnisses an. Das sind alles ausführbare Programmdateien. 

Durch Zusätze in der Kommandozeile können Sie das Ausgabeformat des `ls` Kommandos ändern. Solche Zusätze werden **Optionen** genannt. Die Optionen werden von einem Minuszeichen eingeleitet und bestehen (fast) immer aus einzelnen Buchstaben, die als Schalter bestimmte Eigenschaften des Programms verändern. Alle Optionen des `ls` Kommandos sind [hier](#) beschrieben. Für den Anfang sind die Optionen `-l` (long) für ein ausführliches Listing,  `-a` (all) für die Anzeige aller Dateien sowie `-F` (File) für zusätzliche Symbolisierung bestimmter Dateitypen besonders interessant. Sie können mehrere Optionen hinter einem einzigen Minuszeichen zusammenfassen.

```
$ ls -l /
total 302
drwxr-xr-x    2 ruth      bin           2048 Jun 10 12:39 bin
drwxr-xr-x    2 ruth      ruth          1024 Dec  2 16:46 boot
drwxr-xr-x    2 ruth      ruth          3072 Jul 23 12:33 dev
drwxr-xr-x    8 ruth      bin           2048 Jun 15 12:18 etc
....
-rw-r--r--    1 ruth      ruth          281092 Dec  2 16:54 vmlinuz
$ _
```

## Eigentum und Zugriffsrechte

Als echtes Mehrbenutzerbetriebssystem schützt Linux Ihre privaten Dateien vor dem Zugriff anderer Systembenutzer. Um die Zugriffe auf Dateien und Verzeichnisse zu regulieren werden diesen Eigentümer zugeordnet, die allein über die Zugriffsrechte (Permissions) der anderen Benutzer bestimmen können.

Zur flexibleren Gestaltung dieser Regelung existieren neben dem Eigentümer noch zwei Kategorien für jede Datei: eine Benutzergruppe und die übrigen Anwender.

Jeder Systembenutzer gehört mindestens einer Gruppe an und kann gleichzeitig Mitglied mehrerer Benutzergruppen sein. Indem der Eigentümer die Zugriffsrechte für eine Gruppe und für die übrigen Anwender festlegt, kann er den Kreis der berechtigten Personen für jede Datei und für jedes Verzeichnis individuell bestimmen. Für die Durchsetzung der eingestellten Regeln garantiert das Betriebssystem.

Typ	Eigentümer	Gruppe	Andere	Typ	Eigentümer	Gruppe	Andere
d	r w x	r - x	r - x	-	r w -	r w -	r - -

**Abbildung:** Zugriffsrechte für Datei und Verzeichnis

Die Rechte werden bei einem langen Listing mit `ls -l` (wie im Beispiel oben) im ersten Feld angezeigt. Die erste Stelle gibt den Dateityp an. Der Rest des Feldes besteht aus drei Gruppen zu je drei Stellen. Hier bedeuten:

**r**

(read) Leseberechtigung (bei Verzeichnissen das Recht, den Inhalt des Verzeichnisses anzuzeigen)

**w**

(write) Schreibberechtigung (bei Verzeichnissen das Recht, eine Datei oder ein Unterverzeichnis anzulegen)

**x**

(execute) Ausführberechtigung (bei Verzeichnissen die Möglichkeit, in dieses Verzeichnis zu wechseln)

Die Abbildungen oben zeigen typische Einträge für ein Verzeichnis und eine normale Datei.

Ein Verzeichnis ist durch den Typ `d` gekennzeichnet. In dem Beispiel hat der Eigentümer das Recht, in das Verzeichnis zu schreiben (das heißt Dateien darin anzulegen oder zu löschen), das Verzeichnis aufzulisten und auf die Dateien in dem Verzeichnis zuzugreifen. Die Gruppe und die anderen Systembenutzer können den Inhalt des Verzeichnisses anzeigen und mit `cd` dorthin wechseln, sie können aber keine Datei in dem Verzeichnis anlegen oder eine Datei aus dem Verzeichnis löschen.

Die Datei in dem Beispiel kann von dem Eigentümer und der Gruppe gelesen und beschrieben (verändert) werden, während die anderen Benutzer die Datei nur lesen können.

## Verzeichnisse erstellen und löschen

Mit dem `mkdir` (make directory) Kommando können Sie ein neues Verzeichnis anlegen, vorausgesetzt, Sie haben Schreibberechtigung für das Verzeichnis, in dem das neue Verzeichnis erstellt werden soll. Der Eigentümer des neuen Verzeichnisses sind automatisch Sie selbst.

Das `rmdir` (remove directory) Kommando ist dazu da, Verzeichnisse zu löschen. Es können nur leere Verzeichnisse gelöscht werden.

## Dateien erstellen, verschieben und löschen

Eine Datei wird von einem Programm als Resultat eines bestimmten Prozesses erstellt. Beispielsweise werden die Programme selbst vom C-Compiler gemacht. Textdateien können Sie einfach mit einem Editor wie zum Beispiel `vi`, `joe` oder `emacs` erzeugen oder erweitern.

Es gibt aber auch andere Methoden, Dateien zu erzeugen. Beispielsweise können Sie die Kopie einer existierenden Datei mit dem Befehl `cp` (copy) anfertigen.

Vorausgesetzt, im aktuellen Verzeichnis existiert eine Datei namens `Adressen` und Sie haben Leserecht für die Datei und Schreibberechtigung für das Verzeichnis, erzeugt der Befehl

```
$ cp Adressen Telefonliste
$ _
```

eine genaue Kopie der Datei `Adressen` mit dem Namen `Telefonliste`. Die neue Datei `Telefonliste` ist genauso groß wie `Adressen` und enthält exakt die gleichen Zeichen. Sie belegt zusätzlichen Platz auf der Festplatte, und Veränderungen an der einen Datei sind in der anderen nicht zu sehen.


Das ist nicht selbstverständlich. Mit dem `ln` (link) Kommando können Sie eine andere Art Kopie erzeugen, bei der dieselbe Datei „nur“ einen zusätzlichen Namen erhält. Zum Beispiel

```
$ ln Adressen Adressliste
$ _
```

erzeugt einen neuen Eintrag `Adressliste` im aktuellen Verzeichnis. Das Listing des Verzeichnisses zeigt beide Dateien mit der gleichen Größe, mit den gleichen Zugriffsrechten und mit der gleichen Zeitmarke an. Es **ist** dieselbe Datei, die einfach einen zweiten Namen bekommen hat. Wenn Sie auf die Daten zugreifen wollen, sei es zum Lesen oder Schreiben, macht es keinen Unterschied, unter

welchem Namen Sie die Datei aufrufen. Veränderungen so einer Datei sind automatisch auch unter dem zweiten Namen sichtbar, sobald sie unter dem ersten Namen abgespeichert wurden.

Um eine Datei umzubenennen, können Sie den `mv` (move) Befehl benutzen. Dieser Befehl verändert den Inhalt der Datei nicht. Es wird nur der Name im Verzeichnis geändert.

Zum Löschen von Dateien gibt es das `rm` (remove) Kommando. Eine einmal gelöschte Datei ist verloren. Es gibt keinen Weg, wieder an die Daten gelöschter Dateien heranzukommen.  Deshalb ist beim Umgang mit `rm` immer größte Sorgfalt geboten.

Für das Löschen einer Datei ist übrigens keine Schreibberechtigung für die Datei selbst notwendig. Die Schreibberechtigung auf das Verzeichnis ist hier das entscheidende Kriterium, denn die Löschfunktion (wie auch die zum Umbenennen oder Linken) greift nicht auf die Datei zu, sondern „nur“ auf den Eintrag im Verzeichnis.

## Dateien anzeigen

In der Praxis werden Sie sich häufig Textdateien ansehen wollen. Wenn Sie vorhaben, die Datei zu verändern, werden Sie einen Editor, beispielsweise `elvis`, aufrufen. Um die Datei einfach nur zu lesen, ist ein Editor nicht besonders geeignet. Erstens sind die Editoren zum bloßen Lesen von Textdateien einfach zu aufwendig und zu langsam und zweitens besteht immer die Gefahr der versehentlichen Veränderung einer Textdatei.

Aus diesem Grund gibt es kleine, schnelle Programme, die allein den Inhalt einer Datei anzeigen. Die wichtigsten Programme dieser Gattung sind [more](#) und `less`, sogenannte Pager, mit denen Sie in einer Textdatei „blättern“ können. Das Vorwärtsblättern wird bei beiden Programmen durch ein Leerzeichen (Space) ausgelöst, das Zurückblättern durch den Buchstaben B (back). Das `less`-Programm muß ausdrücklich durch Drücken der Taste Q beendet werden, während `more` automatisch beendet, wenn das Dateiende erreicht wird.

Um eine kurze Datei auf den Bildschirm zu schreiben, kann auch das Programm `cat` verwendet werden.

```
$ cat Adressen
LunetIX      Donaustr. 16          12043 Berlin   030/6235787
FSF          675 Mass Ave                   Cambridge,    MA 02139 USA
XeroxParc    3333 Coyote Hill Raod       Palo Alto,    CA 94304 USA
MIT          545 Technology Square         Cambridge,    MA 02139 USA
O'Reilly     632 Petaluma Avenue            Sebastopol,   CA 95472 USA
$ _
```

schreibt beispielsweise die Datei Adressen auf den Bildschirm. Wenn die Datei nicht auf einen Bildschirm paßt, werden die Zeilen einfach nach oben aus dem Bild geschoben, bis das Dateiende erreicht ist. Sie können den Bildschirm ein Stück zurück- und anschließend wieder vorblättern, indem Sie die BILDHOCH beziehungsweise BILDRUNTER Taste gleichzeitig mit der Shifttaste betätigen.


# Datenströme

Das Verhalten des Programms `cat` zeigt eine Eigenart der Datenverarbeitung unter Linux auf:

Dateien können als eingefrorene „Datenströme“ betrachtet werden. Jeder Druck auf die Tastatur erzeugt ein Zeichen im Datenstrom. Normalerweise wird jedes eingegebene Zeichen auf dem Bildschirm dargestellt, der so ein Echo dieses Eingabedatenstroms ist.

Das `cat`-Kommando öffnet im obigen Beispiel die Datei `Adressen` und leitet die darin enthaltenen Daten als kontinuierlichen Strom auf den Bildschirm.

Passend zum Bild der Datenströme können Sie sich verschiedene Kanäle vorstellen, in denen diese Ströme wie Wasser fließen. Linux bietet sehr flexible Möglichkeiten, diese Ströme zu steuern. Damit nicht jeder Strom ausdrücklich gesteuert werden muß, gibt es einige Standardkanäle: die Standardeingabe, die Standardausgabe und die Standardfehlerausgabe.

Die Standarddatenkanäle sind immer offen. Normalerweise ist der Bildschirm die Standardausgabe und gleichzeitig auch die Standardfehlerausgabe.  Die Standardeingabe ist, wie zu erwarten war, normalerweise die Tastatur. Die drei Standardkanäle sind numeriert: die Standardeingabe hat die Nummer Null, die Standardausgabe und die Standardfehlerausgabe haben die Nummern Eins und Zwei.

Das `cat`-Kommando schreibt normalerweise auf die Standardausgabe. Durch die Operatoren `>` und `<` im Kommandozeileninterpreter (der Shell) können die Datenströme aber auch umgelenkt werden. Zum Beispiel

```
$ cat Adressen > Telefonliste
$ _
```

leitet den Datenstrom, den `cat` aus der Datei `Adressen` liest und in die Standardausgabe schreibt, in die Datei `Telefonliste` um. Die `Telefonliste` speichert den Datenstrom wieder; es entsteht also eine Kopie der Datei `Adressen`. Das gleiche Ergebnis hätten Sie natürlich auch mit dem `cp`-Kommando erzielen können.

Umgekehrt kann auch der Eingabekanal umgelenkt werden:

```
$ sort < Adressen
FSF          675 Mass Ave          Cambridge,    MA 02139 USA
LunetIX      Donaust. 16          12043 Berlin 030/6235787
MIT          545 Technology Square Cambridge,    MA 02139 USA
O'Reilly     632 Petaluma Avenue   Sebastopol,  CA 95472 USA
XeroxParc    3333 Coyote Hill Road Palo Alto,    CA 94304 USA
$ _
```

gibt beispielsweise die Adressen alphabetisch sortiert auf dem Bildschirm aus.

# Verknüpfung mehrerer Kommandos: Pipelines

Um in der Bilderwelt von Datenströmen zu bleiben, bietet Linux zu den offenen Kanälen noch Röhren, sogenannte Pipelines. Damit wird der Ausgabekanal eines Kommandos direkt in den Eingabekanal eines anderen Kommandos geleitet.

Der Operator für diese Pipeline ist der senkrechte Strich |.

Auf der deutschen Tastatur wird er durch gemeinsames Drücken der rechten ALT mit der < Taste (links unten neben der Leertaste und der linken ALT Taste) erzeugt.

```
$ sort Adressen | uniq | less
[Ausgabe von less]
$ _
```

ist so eine Pipeline aus gleich drei Kommandos. Das sort Kommando liest die Datei Adressen (die Eingabeumleitung aus dem früheren Beispiel ist nicht notwendig) und sortiert deren Inhalt zeilenweise. Die sortierten Zeilen werden von der Standardausgabe in eine Pipeline gespeist und so dem Programm uniq zugeführt, das die Zeilen vergleicht und alle Zeilen entfernt, die doppelt vorkommen. Die Ausgabe von uniq wird wiederum in die zweite Pipeline gespeist und schließlich dem less-Programm zugeführt, das die sortierte und von den Doppeln befreite Adreßdatei bildschirmweise anzeigt.



Mit diesen Röhren lassen sich mit den vielseitigen Kommandos des Linux-Basissystems auf einfache Weise auch komplizierte Aufgaben schnell und zweckmäßig lösen. Die kleinen hochspezialisierten Programme aus der ``Werkzeugkiste" (Toolbox), die als Glieder in einer Kette von Pipelines verwendet werden können, werden auch als **Filter** bezeichnet.

## Eingabehilfen von der Shell

Der Kommandozeileninterpreter, die Shell, ist die Benutzeroberfläche von Linux. Es gibt auch eine ausgezeichnete grafische Benutzeroberfläche - das X Window System - aber auch hier hat die normale Shell eine herausragende Bedeutung.

Dem DOS-geschädigten Anwender mag das seltsam anachronistisch vorkommen, gibt es dort doch schon seit Jahren erfolgreiche Tools und Commanders, die die Berührung mit dem „rohen" Kommandozeileninterpreter vermeiden helfen. Da scheint Linux mit seinen Shells reichlich veraltet.

Wenn Sie aber den Umgang mit einer Shell wie der bash, der tcsh oder der zsh einmal gelernt und sich an die vielfältigen Hilfen und Möglichkeiten gewöhnt haben, werden auch Sie diese großen Shells den Kommandeuren und ihren Artgenossen vorziehen.

Zum Beispiel bieten alle modernen Shells einen Kommandozeilenspeicher. In der bash, der Standardshell der meisten Linux-Distributionen, können Sie mit den Cursortasten HOCH und RUNTER in der eigenen Computergeschichte nach Wiederverwendbarem herumstöbern. Hier stehen die letzten hundert  Kommandozeilen zur Auswahl. Auch die gezielte Suche nach einer bestimmten Kommandozeile ist möglich, wenn Sie ein eindeutiges Wort aus dieser Zeile erinnern: Mit der Tastenkombination CONTROL-R  erscheint bei der bash anstelle des bekannten Promptes die

Aufforderung, den Suchbegriff einzugeben. Mit jedem eingegebenen Zeichen wird weiter rückwärts im Kommandozeilenspeicher nach einem passenden Begriff gesucht und die erste passende Zeile sofort angezeigt. Wenn die gewünschte Zeile auf diesem Weg gefunden wurde, können Sie mit den übrigen Editorfunktionen (siehe unten) die Zeile nochmal bearbeiten und schließlich mit einem Zeilenende `RETURN` abschließen.

Zu dem Kommandozeilenspeicher bieten die modernen Shells regelrechte Kommandozeileneditoren, mit denen eine Kommandozeile auf vielfältige Weise bearbeitet werden kann. Mit den Cursortasten kann die Einfügemarke an eine beliebige Stelle gebracht werden, die `BACKSPACE` und `DELETE` Tasten löschen jeweils ein Zeichen. Die weiteren Editorbefehle sind an die Befehlssätze von den Standardeditoren `vi` oder `emacs` angelehnt.

Eine der mächtigsten Funktionen ist die automatische Ergänzung einzelner Wörter durch die Shell. Mit der Tabulatortaste wird diese Funktion bei der `bash` ausgelöst. Die Shell versucht, das bis dahin eingegebene Wort zu ergänzen, indem sie das erste Wort einer Kommandozeile mit allen möglichen Kommandos, die folgenden Wörter mit Datei- und Verzeichnisnamen vergleicht und die längste eindeutige Lösung in die Kommandozeile schreibt. Bei geschickter Benennung Ihrer Dateien reicht es häufig aus, wenn Sie zwei oder drei Buchstaben eines Dateinamens eingeben, der Rest wird von der Shell automatisch ergänzt.

Eine weitere Hilfe, die die Shell beim Eingeben von Dateinamen anbietet, sind die sogenannten „Jokerzeichen“, „Wildcards“ oder „regulären Ausdrücke“, mit denen mehrere Dateien mit gleichen Namensteilen in einem einzigen Ausdruck angegeben werden können. Beispielsweise

```
$ ls -l /bin/m*
-rwxr-xr-x  1 ruth      other      4104 Sep  4 1992 /bin/mkdir
-rwx--x--x  1 ruth      ruth        5212 Sep  5 17:27 /bin/mkfs
-rwxr-xr-x  1 ruth      other      3336 Sep  4 1992 /bin/mknod
-rwx--x--x  1 ruth      ruth      12220 Apr 17 1993 /bin/mount
-rwxr-xr-x  1 ruth      other      6724 Sep  4 1992 /bin/mv
$ _
```


gibt ein ausführliches Listing aller Kommandos im Verzeichnis `/bin` aus, deren Name mit dem Buchstaben `m` beginnt.

Weitere Informationen zur `bash` finden Sie [hier](#).

## Virtuelle Terminals und Hintergrundprozesse

Damit Sie die Multitaskingfähigkeit von Linux auch als einzelner Benutzer ausschöpfen können, bietet Linux verschiedene Möglichkeiten, mehrere Programme nebeneinander laufen zu lassen.

Die erste dieser Möglichkeiten wird durch die sogenannten virtuellen Terminals realisiert. Obwohl Sie nur eine einzige Bildschirm- Tastaturkombination, die sogenannte Console, an Ihrem PC angeschlossen haben, können Sie mit Linux arbeiten als säßen Sie an mehreren Rechnern gleichzeitig. Die reale Console wird durch Linux zu mehreren logischen Terminals, zwischen denen durch die Tastenkombination `ALT-F1` bis `ALT-F8` umgeschaltet werden kann. Auf einigen dieser virtuellen

Terminals wird ein eigener Loginprompt ausgegeben.  Damit erhalten Sie die Möglichkeit, sich gleichzeitig mehrfach einzuloggen, eventuell sogar unter verschiedenen Benutzernamen (wenn Ihnen mehrere Accounts auf dem Rechner gehören).



Auf jedem virtuellen Terminal können Programme gestartet werden, die auch weiterlaufen, wenn auf ein anderes Terminal umgeschaltet wird. Die Tastatureingabe wird immer nur an das aktuelle Terminal ``im Vordergrund" weitergegeben. Die Bildschirmausgabe der einzelnen Terminals wird erst sichtbar, wenn dorthin umgeschaltet wurde. Nur die Statusmeldungen und Fehlerausgaben des Kernels werden nicht auf ein virtuelles Terminal, sondern auf die Console direkt ausgegeben und erscheinen dadurch immer auf dem Bildschirm.

Die zweite Möglichkeit, die Multitaskingfähigkeit von Linux zu nutzen, besteht darin, Prozesse auf einem Terminal in der aktuellen Shell im Hintergrund zu starten. Das geschieht, indem Sie am Ende des Kommandos ein Ampersand `&' eingeben. So ein Kommando wird ganz normal gestartet, die Shell wartet aber nicht auf dessen Beendigung, sondern gibt sofort wieder den Prompt aus und wartet auf Ihre nächste Eingabe. Je nachdem, wie das Terminal eingestellt ist ([siehe stty](#)), erscheinen die Meldungen des Hintergrundprogramms auf dem Bildschirm, oder das Hintergrundprogramm wird für die Ausgabe angehalten, bis es wieder in den Vordergrund geholt wird. Durch Umleitung des Standardausgabekanals in eine Datei kann die Ausgabe eines Hintergrundprogramms auch für spätere Bearbeitung gespeichert werden.

```
$ sort Adressen | uniq > Adressliste &
[1] 19365
$ _
```

bearbeitet beispielsweise die Adreßdatei wie im Beispiel oben, schreibt die sortierte Liste aber in eine Datei, anstatt sie auf dem Bildschirm auszugeben. Die Zahlen, die nach dem Abschluß der Kommandozeile erscheinen, werden von der Shell ausgegeben und teilen Ihnen erstens die Jobnummer und zweitens die Prozeßnummer des gerade im Hintergrund gestarteten Prozesses mit.

Es gibt noch eine dritte Möglichkeit, Prozesse in den Hintergrund zu bringen. Mit der Tastenkombination `CONTROL-Z` können Sie ein im Vordergrund laufendes Programm anhalten. Die Shell zeigt wie oben die Jobnummer und zusätzlich den Status (`Stopped`) und den Namen des angehaltenen Prozesses an und gibt schließlich wieder eine Eingabeaufforderung aus. Mit den sogenannten Jobcontrol Funktionen der Shell können Sie ein angehaltenes Kommando im Hintergrund oder im Vordergrund weiterlaufen lassen.

Wenn beispielsweise die laufende `sort` Pipeline mit `CONTROL-Z` angehalten wird, kann sie mit dem `bg`-Kommando im Hintergrund (`background`) fortgesetzt werden...

```
$ sort Adressen | uniq >Adressliste

[1]+  Stopped                  sort Adressen | uniq >Adressliste
$ bg %1
[1]+ sort Adressen | uniq >Adressliste &
$ _
```

...das Shellkommando [fg](#) setzt den gleichen Job im Vordergrund fort.

```
$ fg %1
sort Adressen | uniq >Adressliste
$ _
```




**Next:** [Zahlensysteme](#) **Up:** [Grundlagen](#) **Previous:** [Das Betriebssystem](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

# Zahlensysteme

Die internen Abläufe aller Computer sind in Speichereinheiten, sogenannten Bytes organisiert. Um Daten aus der realen Welt mit einem Computer bearbeiten zu können, müssen sie im Arbeitsspeicher vorliegen. Dazu müssen sie so dargestellt werden, daß sie in die Speichereinheiten passen.

Die in der Regel analogen Daten der realen Welt werden für ihre Repräsentation im Computer „digitalisiert“ oder „codiert“.  Zur Codierung finden verschiedene Modelle Verwendung.

Als Formate für die digitale Darstellung von Daten sind vor allem fünf Typen von Bedeutung:

## Binärzahl

Ein Byte besteht auf den meisten Computern aus acht Bits, die jeweils gesetzt oder ungesetzt sein können. Diesen beiden Zuständen kann einfach jeweils eine Zahl zugeordnet werden - die 1 für gesetzt und die 0 für ungesetzt - und damit ein Byte als eine achtstellige Binärzahl dargestellt werden. Eine achtstellige Binärzahl kann  $2^8 = 256$  verschiedene Kombinationen von Nullen und Einsen enthalten und damit beispielsweise von 0 bis (dezimal) 255 zählen. Mit diesen Zahlen läßt sich rechnen wie mit den „normalen“ Dezimalzahlen. Dieser faszinierenden Eigenschaft und der booleschen Algebra verdanken wir erst die Möglichkeit, Computer zum Rechnen zu benutzen.

Jede natürliche Zahl läßt sich im binären Zahlensystem darstellen.

## Zeichen

Die modernen Computer werden zu einem großen Teil nicht zum Berechnen von Zahlen, sondern zum Bearbeiten von allgemeineren Daten benutzt. Die meisten dieser Daten sind Texte irgendeiner Art. Daher gibt es schon seit den frühesten Anfängen der elektronischen Datenverarbeitung Tabellen, die den verschiedenen Bytes Buchstaben und andere Zeichen zuordnen. Die wichtigste dieser Tabellen ist der „American Standard for Coded Information Interchange“, die sogenannte ASCII-Tabelle. In dieser Tabelle werden 128 Zeichen verschiedenen Bytes zugeordnet. Das höchste Bit ist bei Computern mit acht Bits pro Byte immer null. Neben dem Alphabet werden in dieser Tabelle eine ganze Reihe von Satz- und Sonderzeichen, sowie Zeichen zur Terminalsteuerung festgelegt.

Internationale Zeichen, wie zum Beispiel die deutschen Umlaute, sind in dieser Tabelle nicht enthalten. Für die internationalen Zeichensätze gibt es verschiedene Tabellen. Unter Linux sind zwei Tabellen von besonderer Bedeutung:

1.  
Der PC-Zeichensatz (ANSI) ist fest im ROM der Grafikkarte gespeichert. Die Buchstaben und Grafikzeichen auf dem Textbildschirm stammen aus diesem Zeichensatz.
2.  
Im ISO-Latin-1 Zeichensatz sind herstellerunabhängig die nationalen Sonderzeichen für

die Länder Mittel- und Südeuropas enthalten.

Sie finden eine Tabelle mit beiden Zeichensätzen [im Anhang](#).

## Dezimalzahl

Wahrscheinlich wegen der passenden Anzahl Finger rechnen wir normalerweise im Dezimalsystem. Dieses Zahlensystem benutzt alle Ziffern von 0 bis 9.

Wie schon gesagt, kann ein Byte durch eine Dezimalzahl dargestellt werden, indem alle möglichen Bytes aus je acht Bits von 0 bis 255 abgezählt werden. Es ist aber ebenso möglich, das höchste Bit eines Bytes als Vorzeichen zu interpretieren. Damit wird die Menge aller Bytes auf die Zahlen von -128 bis 127 abgebildet. Bei der Darstellung von Binärzahlen beziehungsweise Bytes durch Dezimalzahlen muß also immer darauf geachtet werden, ob die Dezimalzahlen mit einem Vorzeichen behaftet sind oder nicht.

Die Darstellung von Dezimalzahlen größer als 255 bzw. kleiner als -128 im Computer wird durch die Verwendung mehrerer Bytes für eine Zahl realisiert. Mit zwei Bytes können kleine natürliche Dezimalzahlen (short integer) von 0 bis 65535 bzw. ganze Zahlen von -32768 bis 32767 dargestellt werden. Größere Zahlen müssen durch entsprechend mehr Bytes codiert werden. Der GNU-C-Compiler von Linux bietet standardmäßig vier Byte große *long integer* an.

Als annäherndes Modell für rationale oder reelle Zahlen benutzen Computer Fließkommazahlen. Eine Fließkommazahl mit einfacher Genauigkeit (float) wird intern durch vier Bytes (=32 Bits) dargestellt, von denen ein Bit das Vorzeichen, 8 Bit den Exponenten mit eigenem Vorzeichen und die restlichen 23 Bit die Mantisse darstellen. Fließkommazahlen mit doppelter Genauigkeit (double) werden unter Linux normalerweise durch acht Bytes mit 11 Bit Exponenten und 52 Bit Mantisse dargestellt. Transzendente Zahlen können vom Computer nicht dargestellt und deshalb auch nicht bearbeitet werden. Rationale Zahlen, die außerhalb des hier dargestellten Bereichs liegen, können unter Linux nicht mit Standardfunktionen berechnet werden.

Mit speziellen Funktionen können natürlich von einem Computer weit größerer Genauigkeiten erreicht werden. Indem immer mehr Bytes zur Darstellung einer Zahl herangezogen werden lassen sich reelle Zahlen theoretisch beliebig weit annähern. Allein der verfügbare Speicherplatz begrenzt diese Beliebigkeit.

## Oktalzahl

Genauso, wie es möglich ist, ein Zahlensystem auf den Ziffern 0 und 1 aufzubauen, kann man auch auf anderen Ziffernfolgen Zahlensysteme aufbauen. Die Zahlen aus dem Zahlensystem zur Basis 8 heißen Oktalzahlen. Die Oktalzahlen benutzen nur die Ziffern von 0 bis 7. Genau wie im Dezimalsystem werden beim Zählen die Ziffern der niedrigsten Stelle solange erhöht, bis die höchste Ziffer erreicht ist, in diesem Fall also die 7, und danach wird die nächsthöhere Stelle um eins erhöht und so weiter. Damit ist die Oktalzahl 10 identisch mit der Dezimalzahl 8.

Der Vorteil des oktal Zahlensystems besteht in der leichten Umrechenbarkeit von Oktalzahlen in Binärzahlen und umgekehrt. Weil die Oktalzahlen „handlicher“ sind als ihre binären Äquivalente, werden sie gern zur Darstellung von Bytes verwendet.

0	=	000
1	=	001
2	=	010
3	=	011
4	=	100
5	=	101
6	=	110
7	=	111

	1	1	1	1	0	1	0	1
0	1	1	1	1	0	1	0	1
	3		6			5		

## Hexadezimalzahl

Es gibt in unserem Kulturkreis nur 10 gebräuchliche Ziffern. Aber es gibt eigentlich keinen Grund, auf Zahlensysteme mit mehr Ziffern zu verzichten. Das im Computerbereich gebräuchliche Hexadezimalsystem benutzt 16 Ziffern. Zusätzlich zu den Ziffern 0 bis 9 werden hier die Buchstaben a bis f als elfte bis sechzehnte Ziffer benutzt. Dieses Zahlensystem erlaubt auch eine relativ einfache Umrechnung von Binärzahlen in Hexadezimalzahlen und umgekehrt. Hierzu werden in der gleichen Weise wie bei den Oktalzahlen die Hexadezimalziffern in vierstellige Binärzahlen und Gruppen zu je vier Binärziffern zu Hexadezimalziffern gewandelt.

1	1	1	1	0	1	0	1
1	1	1	1	0	1	0	1
	F				5		

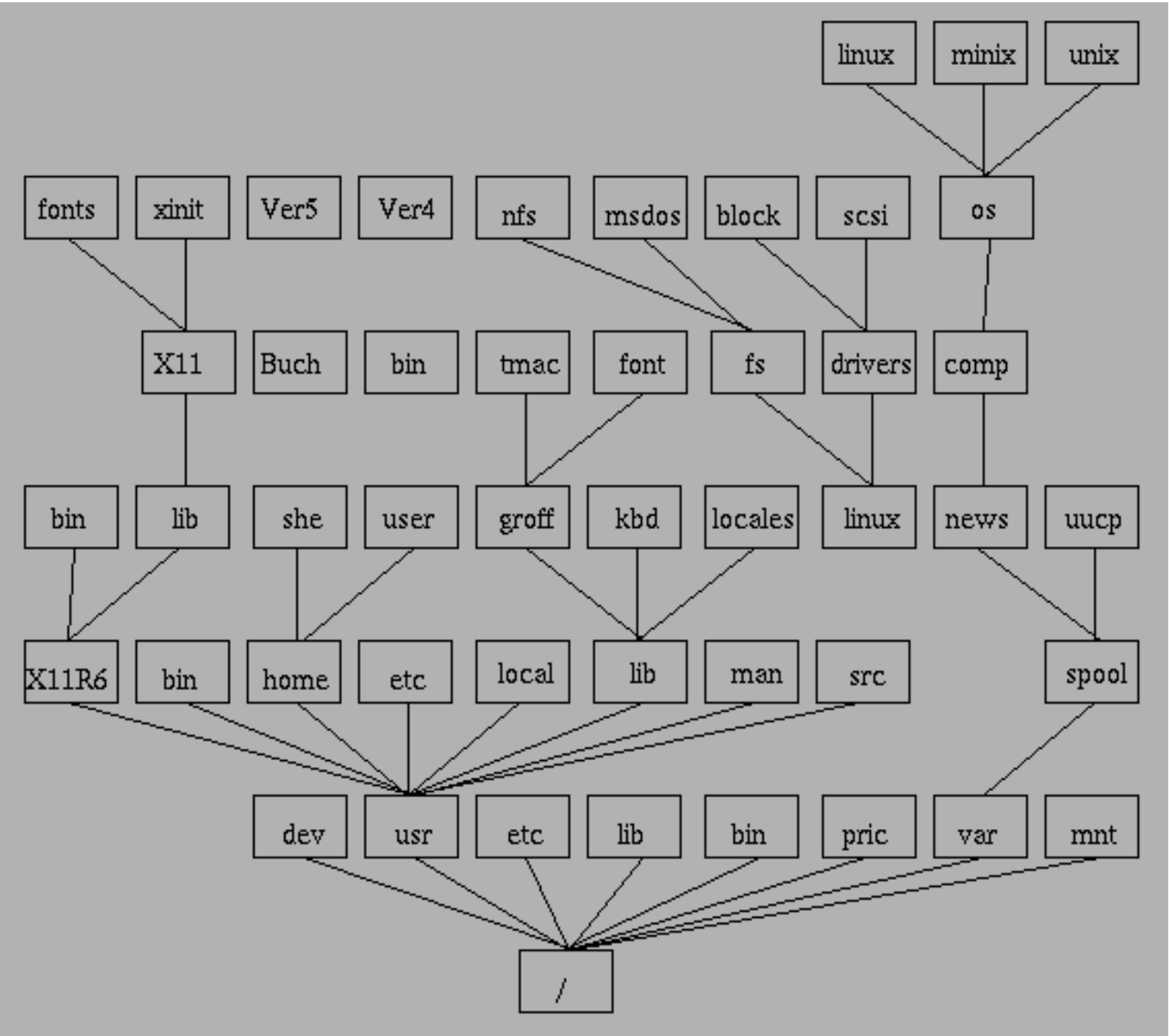
Im Hexadezimalsystem entspricht ein Byte aus acht Bits genau einer zweistelligen Hexadezimalzahl.

Es ist üblich, Zahlen mit einer führenden Null als Oktalzahl zu interpretieren und Hexadezimalzahlen durch den Präfix 0x zu kennzeichnen. Wenn aus dem Zusammenhang eines Textes nicht deutlich wird, in welchem Zahlensystem gerechnet wird, kann durch eine kleine tiefgestellte Zahl die entsprechende Zahlenbasis angegeben werden.

[Next](#)
[Up](#)
[Previous](#)
[Contents](#)
[Index](#)

Next: [Reise durch's Dateisystem](#) Up: [Grundlagen](#) Previous: [Die ersten Schritte](#)

# Reise durch's Dateisystem



**Abbildung:** Verzeichnisbaum (unvollständig)

Die meisten Linux-Distributionen kommen heute auf einer CD-ROM und belegen nach der Installation über 100 Megabyte auf der Festplatte. Um mit solch einer Menge Daten überhaupt arbeiten zu können, müssen Sie sich einen groben Überblick über die Ordnung und die Funktion der Dateien verschaffen. Zu diesem Zweck wird in diesem Kapitel eine Rundreise durch das Dateisystem veranstaltet, auf der Ihnen die wichtigsten Verzeichnisse und Dateien vorgestellt werden.

Dieses Kapitel soll nicht wirklich Seite für Seite durchgelesen werden. Um bei dem Bild der Reise zu bleiben: Sie sollten zuerst einmal durchreisen um sich einen Überblick zu verschaffen. Später können

Sie dann einzelne Ziele auswählen und dort mit ausführlicheren Studien beginnen.

Die Struktur des Linux-Dateisystems orientiert sich grundsätzlich an Unix. Es gibt jedoch auch bei den verschiedenen Unix-Versionen unterschiedliche Modelle, die bei den frühen Linux-Distributionen ohne klares Konzept zusammengewürfelt worden sind.

Um ein einheitliches Dateisystem für alle Linux-Distributionen einzuführen, ist ein Standard für das Dateisystem ausgearbeitet und ausführlich diskutiert worden. Dieser Standard ist stark an System V Release 4 orientiert. Er ist aber keine bloße Adaption eines bestehenden Layouts, sondern ihm liegen einige als wichtig erkannte Prinzipien zugrunde. Obwohl es keine Instanz gibt, die die Verwendung des Standard-Dateisystems vorschreibt, hat es sich bei allen Distributionen durchgesetzt.

Wenn Sie eine Linux-Distribution installiert haben, die dem Dateisystemstandard (noch) nicht folgt, werden Sie einige der hier vorgestellten Dateien an anderen Orten und einige Verzeichnisse gar nicht finden. Trotzdem werden Sie viele Übereinstimmungen feststellen. Die Funktion und das Format der Dateien ändert sich natürlich mit dem bloßen Layout des Dateisystems nicht.

- 
- [Konzepte](#)
    - [Der File-System-Standard](#)
  - [Rootpartition und Wurzelverzeichnis](#)
    - [Das Verzeichnis /boot](#)
    - [Das Verzeichnis lost+found](#)
    - [Das Verzeichnis /mnt](#)
    - [Das Verzeichnis /root](#)
  - [Die Binärverzeichnisse](#)
  - [Die Gerätedateien im Verzeichnis /dev](#)
    - [Der Arbeitsspeicher](#)
    - [Runde Scheiben](#)
    - [Zeichenorientierte Ein- und Ausgabegeräte](#)
    - [Daten am laufenden Band](#)
  - [Die Konfigurationsdateien im Verzeichnis /etc](#)
    - [/etc/adjtime](#)
    - [/etc/fdprm](#)
    - [/etc/fstab](#)
    - [/etc/gettydefs](#)
    - [/etc/group](#)
    - [/etc/hosts](#)
    - [/etc/inittab](#)
    - [/etc/issue](#)

- [/etc/ld.so.conf](#)
- [/etc/login.defs](#)
- [/etc/magic](#)
- [/etc/man.config](#)
- [/etc/motd](#)
- [/etc/nologin](#)
- [/etc/passwd](#)
- [/etc/printcap](#)
- [/etc/profile](#)
- [/etc/psdatabase](#)
- [/etc/rc\\*](#)
- [/etc/securetty](#)
- [/etc/shells](#)
- [/etc/syslogd.conf](#)
- [/etc/termcap](#)
- [Nichts über TCP/IP](#)
- [Home, Sweet Home](#)
  - [Das möblierte Zimmer](#)
- [Die Bibliotheken in ./lib](#)
  - [Compiler, Bibliotheken und Daten in /usr/lib](#)
- [Erweiterungspakete in /opt](#)
- [Die Prozeßdaten im Verzeichnis /proc](#)
- [Die temporären Dateien im Verzeichnis ./tmp](#)
- [Das Verzeichnis /usr](#)
  - [/usr/X11R6](#)
  - [/usr/dict](#)
  - [/usr/doc](#)
  - [/usr/games](#)
  - [/usr/include](#)
  - [/usr/local](#)
  - [/usr/info](#)
  - [/usr/man](#)
  - [/usr/share](#)
  - [/usr/src](#)
  - [/usr/spool](#)

- [Das Verzeichnis /var](#)

---

Next	Up	Previous	Contents	Index
------	----	----------	----------	-------

**Next:** [Konzepte](#) **Up:** [Das Linux Anwenderhandbuch](#) **Previous:** [Zahlensysteme](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)



**Next:** [Rootpartition und Wurzelverzeichnis](#) **Up:** [Reise durch's Dateisystem](#) **Previous:** [Reise durch's Dateisystem](#)

## Subsections

- [Der File-System-Standard](#)

---

# Konzepte

Vor Antritt der Reise sollten Sie einige grundlegende Konzepte des Linux-Dateisystems verstehen.

Wahrscheinlich kennen Sie von MS-DOS Festplattenpartitionen als „Laufwerke“ C:\> oder D:\>. Hier müssen Sie als Benutzer das „richtige“ Laufwerk wählen, wenn Sie ein bestimmtes Verzeichnis oder eine bestimmte Datei erreichen wollen.

Wenn Sie die Linux-Distribution selbst installiert haben, werden Sie gesehen haben, daß auch Linux mehrere Festplattenpartitionen benutzen kann. Bei einem fertig installierten Linux-System brauchen Sie sich aber um solche Hardwareangelegenheiten nicht mehr zu kümmern: Es ist eines der wichtigsten Konzepte des Linux-Dateisystems, die hardwareabhängigen Aspekte Ihres Arbeitssystems vollständig zu verbergen.

Auf der Benutzerebene bewegen Sie sich in einem einzigen, hierarchischen Dateisystem, in das alle Datenträger integriert sind. Es ist wie ein Baum aufgebaut, dessen Äste und Zweige die Verzeichnisse und dessen Blätter die Dateien sind. Eine bestimmte Festplattenpartition - die Rootpartition - wird automatisch vom Kernel als „Wurzel“ des Dateisystems („Rootfilesystem“) aktiviert; andere Partitionen werden bei der Systeminitialisierung auf bestimmte Verzeichnisse in dieser Rootpartition aufgesetzt. Für Sie als Benutzer sind die „Nahtstellen“ zwischen den Partitionen nur durch spezielle Hilfsprogramme sichtbar, aber für Ihre Arbeit mit dem System sind sie ohne Bedeutung.

Die Funktion von Verzeichnissen oder *Directories* wird Ihnen wahrscheinlich bekannt sein: Sie können sie sich als eine Art Behälter vorstellen, in denen Dateien und/oder weitere Verzeichnisse enthalten sein können.

Dateien (*Files*) sind auf den ersten Blick Texte, Programme oder sonstige Daten, die in ihrer maschinenlesbaren Form als „Strom“ von Bytes in Blöcken auf einem Massenspeicher (Festplatte oder Diskette) festgehalten und von dort wieder gelesen werden können. Dateien werden bei der Erzeugung mit einem Namen in ein Verzeichnis eingetragen und können danach unter diesem Namen angesprochen werden.

Darin unterscheidet sich das Linux-Dateisystem noch nicht von dem anderer Betriebssysteme. Erst bei näherer Betrachtung ändert sich das Bild: bei Linux können oder müssen grob sechs Arten von Dateien unterschieden werden.

## Normale Dateien

(*regular files*) entsprechen dem oben dargestellten Bild. Sie können ebenso gut lesbare Gedichte

wie ausführbare Maschinenprogramme enthalten. Sie belegen normale Datenblöcke auf der Festplatte.

## Verzeichnisse

(*directories*) sind, genauer betrachtet, Dateien, in denen auf eine spezielle Art weitere Dateien „enthalten“ sind. Auch sie belegen Datenblöcke auf der Festplatte. Der Aufbau dieser Spezialdateien wird im Kapitel über die [Dateisysteme](#) erläutert.

## Gerätedateien

sind Bindeglieder zwischen den Hardwarekomponenten und Geräten (*devices*) am Rechner bzw. den Gerätetreibern im Kernel auf der einen Seite und der Software im Laufzeitsystem, also den Anwenderprogrammen, auf der anderen. Alle Benutzerprozesse, die auf ein angeschlossenes Gerät oder eine Hardwarekomponente zugreifen wollen, müssen das über eine solche Gerätedatei tun.

## Sockets

*Sockets* sind Spezialdateien aus dem Bereich der TCP/IP-Vernetzung, mit denen der Datenaustausch zwischen zwei lokal laufenden Prozessen über das Dateisystem realisiert werden kann.


## FIFOs

(named *pipes*) stellen eine zweite, einfachere Methode des Datentransports zwischen zwei Prozessen über das Dateisystem dar. Wie die Pipelines in der Shell können die FIFOs Daten nur in einer Richtung transportieren.

## Links

sind zusätzliche Namen (Verzeichniseinträge) für existierende Dateien. Es werden symbolische Links und Hardlinks unterschieden. Während Hardlinks vollkommen gleichwertige Verzeichniseinträge für eine existierende Datei sind, bestehen symbolische Links aus einer Spezialdatei, deren Inhalt ein Zeiger auf eine andere Datei ist.

# Der File-System-Standard

Der File-System-Standard ist das Projekt einer Gruppe von Linux-Systemadministratoren, -Entwicklern und -Benutzern, die ihre Erfahrungen und Vorstellungen ausführlich im Internet ausgetauscht und diskutiert haben. Das Ergebnis ist von Daniel Quinlan (quinlan@bucknell.edu) im File-System-Standard zusammengefaßt worden, dessen Version 1.2 im März 1995 fertiggestellt wurde.  Die Arbeit an der Standardisierung des Dateisystems geht weiter: zur Zeit werden Empfehlungen für ein gemeinsames Layout der Verzeichnisstruktur von BSD und Linux erarbeitet. Der aktuelle Entwurf ist unter dem Namen *Filesystem Hierarchy Standard -- Draft 7* ebenfalls per FTP im Internet zu finden.

Mehrere zentrale Prinzipien ziehen sich durch den Standard:

- Um der zunehmenden Bedeutung von Linux als verteiltes Betriebssystem auf lokalen TCP/IP-Netzwerken Rechnung zu tragen, muß ein maximaler Anteil des Dateisystems für die gemeinsame Nutzung (über NFS) vorbereitet sein. Eine Voraussetzung dafür ist, daß dieser Teil des Dateisystems beim normalen Betrieb nicht verändert wird.

Dieses Prinzip führt zu einer Unterscheidung der Daten in zwei Richtungen:

- verteilte Daten / lokale Daten
- variable Daten / statische Daten
- Die Unterteilung des Dateisystems sollte nach den bei Unix üblichen funktionalen Kategorien stattfinden. Das bedeutet, daß jeweils Binärdateien, Konfigurationsdateien, Hilfstexte und sonstige Daten in verschiedenen Verzeichnissen untergebracht werden.
- Die beiden Prinzipien oben führen insbesondere zur Forderung nach einer klaren und übersichtlichen Strukturierung der Konfigurationsdateien. In dieser Struktur muß sich auch die Unterscheidung zwischen netzweit verteilten und (rechner-) lokalen Daten sowie die Unterscheidung zwischen der von einer Distribution eingespielten und der zusätzlich (netz-) lokal hinzugefügten Software widerspiegeln.

An dieser Stelle beginnt die eigentliche Reise durch das Dateisystem. Auch wenn Ihnen an vielen Stellen ein Blick hinter die Kulissen gewährt wird, sehen Sie auf der Oberfläche ein leicht idealisiertes Dateisystem. Damit wird Ihnen nichts Wichtiges vorenthalten, sondern im Gegenteil Ihr Blick auf das Wesentliche gerichtet.

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [Rootpartition und Wurzelverzeichnis](#) **Up:** [Reise durch's Dateisystem](#) **Previous:** [Reise durch's Dateisystem](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

## Subsections

- [Das Verzeichnis /boot](#)
- [Das Verzeichnis lost+found](#)
- [Das Verzeichnis /mnt](#)
- [Das Verzeichnis /root](#)


# Rootpartition und Wurzelverzeichnis

Wie bereits gesagt, bietet das Linux-Dateisystem das Bild eines Baumes mit einer Wurzel. Es ist naheliegend, eine Reise durch das Dateisystem an der Wurzel anzufangen.

Ein kurzes Listing zeigt das folgende Bild:

```
$ ls -F /
bin/          etc/          lost+found/   root/         usr/
boot/         home/         mnt/          sbin/         var/
dev/          lib/          proc/         tmp/          vmlinuz
$ _
```

Das Rootfilesystem wird vom Kernel automatisch aktiviert, sobald er in den Arbeitsspeicher geladen und initialisiert ist. Das Wurzelverzeichnis ist von da an fest im Kernel verankert. Der Linux-Kernel erfüllt selbst nur die minimalen Aufgaben eines Betriebssystems. Damit der Kernel auch bestimmte Systemprogramme (wie beispielsweise `init`, `getty` oder eine Shell) ausführen kann, ist diese feste

Verbindung notwendig.  Das bedeutet, daß alle Programme, die vom Kernel unmittelbar beim Systemstart automatisch ausgeführt werden sollen, zusammen mit ihren Konfigurationsdateien im Rootfilesystem angesiedelt sein müssen.

Zusätzlich müssen allein mit dem Rootfilesystem eine Reihe von essentiellen Arbeiten der Systemverwaltung ausführbar sein. Dazu gehören:

- das Erzeugen, Zusammensetzen und Reparieren von Dateisystemen,
- das Sichern und Zurücksichern der Systemdaten sowie die Installation neuer Systemteile und
- in Netzwerksystemen die Herstellung und die Prüfung einer Netzverbindung im lokalen Netz.

Die Größe des Rootfilesystems bzw. die Abgrenzung zum „Benutzersystem“ ist eine der Glaubensfragen beim Design des Dateisystems. Der File-System-Standard sieht die Beschränkung des Rootfilesystems auf das absolut Notwendigste vor. Das hat vor allem den Vorteil, daß das potentielle Risiko eines Datenverlustes durch Fehler im Dateisystem bei einer kleinen Partition, auf der im wesentlichen nur statische Daten gespeichert sind, sehr klein ist. So eine Partition läßt sich auch unter ungünstigen Bedingungen regelmäßig (zum Beispiel auf einer Floppydisk) sichern, so daß die Wiederherstellung des Systems auch nach einem Totalausfall kein ernstes Problem darstellt. Im Idealfall kann eine so angelegte Partition sogar auf einer Bootdiskette Platz finden, so daß das System ohne Festplatte gebootet werden kann (um danach über das lokale Netz zu arbeiten).

Wenn Sie sich das Listing des Wurzelverzeichnis anschauen, werden Sie eine kleine Anzahl Verzeichnisse und eine einzige normale Datei finden. Die Datei kann in Abweichung vom oben gegebenen Beispiel auch `zImage` heißen und enthält den bootfähigen Kernel.

## Das Verzeichnis `/boot`

Das Verzeichnis `/boot` enthält die Dateien des [LILO Bootloaders](#), die nicht ausführbar und auch keine Konfigurationsdateien sind. Das sind in der Regel der gesicherte Master-Boot-Record und die Sector-Map. Hier können auch zusätzliche Kernel-Images abgelegt werden. Nach dem File-System-Standard kann und soll allein der Defaultkernel im Wurzelverzeichnis liegen.

## Das Verzeichnis `lost+found`

In sehr seltenen Fällen kann der Verzeichniseintrag einer Datei gelöscht werden, ohne daß die dazugehörigen Datei aus dem Dateisystem entfernt wird. Diese verlorenen Daten werden bei einem [File-System-Check](#) gefunden und durch einen Eintrag in diesem Verzeichnis wieder zugänglich gemacht. Ein Verzeichnis mit dem Namen `lost+found` finden Sie im Wurzelverzeichnis jeder Partition mit einem EXT2-Dateisystem.

## Das Verzeichnis `/mnt`

Das Verzeichnis `/mnt` beziehungsweise die möglicherweise darin enthaltenen Unterverzeichnisse (`/mnt/cdrom`, `/mnt/floppy` ...) dienen zur vorübergehender Einbindung zusätzlicher Dateisysteme.

## Das Verzeichnis `/root`

Das Verzeichnis `/root` ist optionaler Bestandteil des Standard-Dateisystems und dient als Heimatverzeichnis des Superusers `root`. Obwohl unter diesem Account keine privaten Daten abgelegt werden sollten, dient dieses Heimatverzeichnis zumindest zur Platzierung von Initialisierungsdateien für die Programme, mit denen die Systemverwalterin arbeiten muß.

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [Die Binärverzeichnisse](#) **Up:** [Reise durch's Dateisystem](#) **Previous:** [Konzepte](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

# Die Binärverzeichnisse

Die für Unix typische Strukturierung des Dateisystems nach funktionalen Gesichtspunkten wird bei den Verzeichnissen mit dem Namen `bin` besonders deutlich. Auf den verschiedenen Ebenen des Dateisystems gibt es jeweils Verzeichnisse dieses Namens, denen allen gemeinsam ist, daß sie ausschließlich ausführbare Dateien enthalten.

Der Vorteil dieser Zusammenfassung besteht darin, daß die automatische Suche der Shell (und anderer Programme) nach einem ausführbaren Programm auf diese Weise schnell und unkompliziert ist. Zu diesem Zweck werden alle `bin`-Verzeichnisse mit ihren absoluten Namen in der Umgebungsvariablen [PATH](#) aufgelistet. Die Standardbelegung für `PATH` ist:

```
PATH=/bin:/usr/bin:/usr/X11R6/bin:/usr/local/bin
```

Für die Systemverwalterin kommen noch weitere Verzeichnisse hinzu:

```
PATH=/sbin:/usr/sbin:$PATH
```

Das Verzeichnis `/bin` im Rootfilesystem enthält alle Programmdateien, die für die essentiellen Aufgaben der Systemverwaltung gebraucht werden, die aber auch von den anderen Systembenutzern verwendet werden können.

Im File-System-Standard ist eine Liste der essentiell wichtigen Programmdateien für `/bin` aufgeführt. Natürlich können Sie bei Bedarf weitere Programme in diesem Verzeichnis installieren.

```
$ ls /bin
arch      dd        gzip      more      rmdir     true
bash      df        hostname  mount     sed       umount
cat       dmesg     kill      mv        setserial uname
chgrp     domainname ln         netstat   sh        zcat
chmod     echo      login     ping      stty
chown     ed        ls        ps        su
cp        false     mkdir     pwd       sync
date      gunzip    mknod     rm        tar
$ _
```


Die meisten dieser Kommandos sind im Referenzteil des Buches ausführlich beschrieben.

Das Verzeichnis `/sbin` enthält alle Programmdateien für die essentiellen Aufgaben der Systemverwaltung, deren Ausführung der Superuserin (`root`) vorbehalten ist.

Um das Rootfilesystem möglichst klein zu halten, sind in `/sbin` wie in `/bin` nur die absolut notwendigen Programme enthalten:

```
$ ls /sbin
arp      fdisk      ifconfig    mkfs.minix  sln
badblocks fsck       init        mkfs.xiafs  ssync
clock    fsck.ext2  ldconfig    mklost+found swapoff
```

dumpe2fs	fsck.minix	lilo	mkswap	swapon
e2fsck	fsck.xiafs	mke2fs	reboot	telinit
fastboot	getty	mkfs	route	tune2fs
fasthalt	halt	mkfs.ext2	shutdown	update
\$ _				

Die beiden Binärverzeichnisse im Wurzelverzeichnis haben jeweils ein Pendant im Verzeichnis `/usr`. In `/usr/bin` befindet sich das Gros der Programmdateien einer Distribution. Es muß sich nicht unbedingt um Binärdateien handeln, Shellscripte und andere interpretierbare Textdateien können hier ebenfalls enthalten sein, wenn sie durch die entsprechenden Zugriffsrechte als ausführbar gekennzeichnet sind und der Interpreter durch eine eine Konstruktion  in der ersten Zeile festgelegt ist.

In `/usr/sbin` sind die alle Programme für die Superuserin enthalten, die nicht zum Booten und für die Bearbeitung der Dateisysteme gebraucht werden. Wie das Verzeichnis `/sbin` muß auch `/usr/sbin` im Pfad von Usern ohne Systemverwaltungsaufgaben nicht enthalten sein.

Zwei weitere Sammelverzeichnisse für Binärdateien sind `/usr/local/bin` und `/usr/X11R6/bin`. Das erste enthält Programmdateien, die nicht Bestandteil der Linux-Distribution sind. Das zweite enthält die Programme, die mit der grafischen Benutzeroberfläche von Linux, dem X Window System, arbeiten. Für kommerzielle Systemerweiterungen kann noch das Verzeichnis `/opt/bin` hinzukommen.

---

[Next](#)
[Up](#)
[Previous](#)
[Contents](#)
[Index](#)

**Next:** [Die Gerätedateien im Verzeichnis](#) **Up:** [Reise durch's Dateisystem](#) **Previous:** [Rootpartition und Wurzelverzeichnis](#)

*Das Linux Anwenderhandbuch*  
 (C) 1997 [LunetIX](#)

**Next:** [Die Konfigurationsdateien im Verzeichnis](#) **Up:** [Reise durch's Dateisystem](#) **Previous:** [Die Binärverzeichnisse](#)

## Subsections

- [Der Arbeitsspeicher](#)
    - [Die RAM-Disks /dev/ram\\*](#)
    - [Zufallszahlen aus /dev/random](#)
    - [Die Senke /dev/null](#)
    - [Die Quelle /dev/zero](#)
    - [Die Fehlerquelle /dev/full](#)
    - [Der IO-Bereich /dev/port](#)
    - [Der Arbeitsspeicher /dev/mem und /dev/kmem](#)
  - [Runde Scheiben](#)
    - [Die Diskettenlaufwerke /dev/fd\\*](#)
    - [Die IDE-Festplatten](#)
    - [Die SCSI-Festplatten /dev/sd\\*](#)
    - [Altmetallverwertung: /dev/xd\\*](#)
    - [Das Dateisystem im Dateisystem: /dev/loop?](#)
    - [Zusammenlegung mehrerer Partitionen: /dev/md?](#)
    - [Die CD-ROM-Laufwerke](#)
  - [Zeichenorientierte Ein- und Ausgabegeräte](#)
    - [Die Console und virtuelle Terminals](#)
    - [Pseudoterminals für die grafische Benutzeroberfläche](#)
    - [Die serielle Schnittstelle](#)
    - [Die Busmäuse](#)
    - [Der Wecker: /dev/rtc](#)
    - [Die parallele Druckerschnittstelle /dev/lp?](#)
    - [Die Soundkarte](#)
  - [Daten am laufenden Band](#)
    - [Die QIC-02-Streamer /dev/rmt? und /dev/tape\\*](#)
    - [Die SCSI-Streamer /dev/st? und /dev/nst?](#)
    - [Der Floppystreamer \(ftape\)](#)
-



# Die Gerätedateien im Verzeichnis /dev

Eine der Charakteristika von Linux ist die vollständige Abschirmung der Hardware- von der Benutzerebene. Wann immer ein Anwenderprogramm Daten an ein angeschlossenes Gerät oder eine Hardwarekomponente (Device) schicken oder von dort lesen will, muß es eine Spezialdatei öffnen und die Daten in diese Datei schreiben oder aus dieser Datei lesen.

Die Systemverwalterin hat die gleichen Möglichkeiten, Zugriffsrechte auf die Gerätedateien zu erteilen oder zu verweigern wie bei allen anderen Dateien auch. Indem nur bestimmten Benutzern, beispielsweise Dämonprozessen, der Zugriff auf kritische Gerätedateien erlaubt wird, lassen sich bereits viele Sicherheitsrisiken und Blockadesituationen verhindern. Außerdem kann der Gerätetreiber im Kernel konkurrierende Zugriffe mehrerer Programme auf ein Gerät verhindern.

Ein weiterer entscheidender Vorteil dieses auf den ersten Blick vielleicht etwas restriktiv anmutenden Mechanismus ist die Fähigkeit des Kernels, Prozesse bis zum Eintreten eines Hardwareereignisses in Schlaf zu versetzen. Auf diese Weise kann die Rechenzeit, die sonst in einer Warteschleife für das sogenannte *polling* verschwendet würde, sinnvoll für andere Prozesse verwendet werden.

In dem Verzeichnis /dev sind die „Bilder“ der Hardwarekomponenten und Geräte des Systems abgelegt. Solche Bilder sind keine Dateien im herkömmlichen Sinne. Sie belegen keine Datenblöcke auf der Festplatte, sondern nur I-Nodes. Diese Spezialdateien stellen eine Verbindung zwischen den Anwenderprogrammen und den Gerätetreibern im Kernel her. In die meisten dieser Dateien kann geschrieben oder es kann aus ihnen gelesen werden wie bei normalen Dateien. Der Datenstrom wird vom Kernel übernommen und an das entsprechende Gerät weitergeleitet.

Die Verbindung zum Kernel wird über Slots oder Kanäle hergestellt, die numeriert sind und hinter denen sich die Treiber für die Geräte verbergen. Die Nummer des Gerätetreibers wird als **Hauptgerätenummer** (Major Device Number) bezeichnet. Ein Treiber kann mehrere Geräte des gleichen Typs verwalten. Um die einzelnen Geräte zu unterscheiden, wird dem Treiber eine zweite Zahl, die **Untergerätenummer** (Minor Device Number), übergeben. Diese beiden Zahlen charakterisieren jede Datei im /dev Verzeichnis. Zusätzlich werden noch zwei Arten von Geräten unterschieden: Die blockorientierten Geräte mit direktem Zugriff, wie z. B. Disketten oder Festplatten, und die zeichenorientierten sequentiellen Geräte, wie Drucker, Terminal oder Maus. Damit hat jede Gerätedatei drei „Koordinaten“, mit der sie vom Kernel, unabhängig von ihrem Namen, eindeutig identifiziert werden kann.

Bei einem langen Listing des Verzeichnisses mit `ls -l` werden die Haupt- und Untergerätenummern anstelle der Dateigröße angezeigt. Ob ein Gerät als Zeichen- oder Blockdevice angesprochen wird, kann man am ersten Buchstaben der Zugriffsrechte sehen. Da steht `‘b’` für block- und `‘c’` für zeichenorientierte Gerätedatei.

Die Zahl der vom Linux-Kernel unterstützten Geräte wächst mit der Beliebtheit von Linux ständig an. Die offizielle Liste der registrierten Devices mit ihren Gerätenummern ist bei den Kernelsourcen in der Datei `./linux/Documentation/devices.txt` zu finden.

Bei der Installation einer neuen Linux-Distribution wird durch das Shellsript `MAKEDEV` automatisch ein Satz Gerätedateien erzeugt. In dieser Datei sind die Gerätenummern aller fest im Kernel enthaltenen Treiber gespeichert. Mit `MAKEDEV` lassen sich auch nachträglich Gerätedateien erzeugen. Zusätzliche Gerätedateien mit statischen Gerätenummern können von der Superuserin mit dem [mknod-Kommando](#) angelegt werden.

Die Namensgebung für die Gerätedateien ist Konvention. Jede Datei kann beliebig umbenannt oder durch Links mit anderen Namen verbunden werden, die Benutzung des Gerätes ändert sich dadurch nicht. Der File-System-Standard sieht aber vor, daß bei der Installation einer Linux-Distribution keine symbolischen Links automatisch erzeugt werden sollen.

Bei der Beschreibung der Gerätedateien werden gleichartige Geräte durch Wildcards bezeichnet, wie sie in der Shell verwendet werden. Ein Fragezeichen steht hier für ein einziges Zeichen, ein Asterisk (Stern) steht für mehrere Zeichen. Sie können sich die vollständige Liste der angesprochenen Gerätedateien mit dem `ls`-Kommando anzeigen lassen, indem Sie genau den in der Beschreibung angegebenen Ausdruck als Argument verwenden. Beispielsweise die Bezeichnungen `/dev/fd?` und `/dev/hd*` erweitern sich so:

```
$ ls /dev/fd?
/dev/fd0  /dev/fd1
$ ls /dev/hd*
/dev/hda    /dev/hda3  /dev/hda6  /dev/hdb    /dev/hdb3  /dev/hdb6
/dev/hda1   /dev/hda4  /dev/hda7  /dev/hdb1   /dev/hdb4  /dev/hdb7
/dev/hda2   /dev/hda5  /dev/hda8  /dev/hdb2   /dev/hdb5  /dev/hdb8
$ _
```

Die Gerätedateien sind nicht in dem Sinn wichtig, daß ein User im alltäglichen Betrieb viel mit ihnen zu tun hätte -- jedenfalls nicht bewußt und an der Oberfläche. Die folgenden Beschreibungen sind trotzdem ziemlich ausführlich, um Ihnen ein besseres Verständnis für die Arbeitsweise und die Qualitäten des Betriebssystems zu vermitteln.

## Der Arbeitsspeicher

Der Arbeitsspeicher (RAM) wird unmittelbar von der CPU bearbeitet. Die Organisation dieses Speichers ist eine der Hauptaufgaben des Betriebssystems. Linux erlaubt den Prozessen über spezielle Gerätedateien den Zugriff auf bestimmte Teile des Arbeitsspeichers und bietet Pseudogeräte für bestimmte Zwecke an.

### Die RAM-Disks `/dev/ram*`

Die Dateien `/dev/ram*` bilden die RAM-Disks in das Dateisystem ab, wenn beim Übersetzen des Kernels der Treiber für dieses "Gerät" ausgewählt wurde. Das Betriebssystem stellt 16 RAM-Disks mit jeweils bis zu 4 MB Speicherkapazität bereit.

Unter dem Dateinamen `/dev/initrd` (Minor 250) kann eine zusätzliche RAM-Disk mit Kernelmodulen und einem speziellen Bootsystem zur Initialisierung des Kernels existieren.

Einzelheiten zur Verwendung der RAM-Disks finden Sie im Kapitel über die [Systemverwaltung](#).

### Zufallszahlen aus `/dev/random`

Für viele Computerprogramme ist ein gewisses Maß an Unbestimmtheit zur Erzielung des erwünschten Ergebnisses notwendig. Es gibt Bibliotheksfunktionen, die statistisch gleichverteilte

Pseudozufallszahlen liefern. Damit lassen sich beispielsweise eindeutige Namen für temporäre Dateien generieren. Diese Pseudozufallszahlen sind jedoch vorhersehbar: Wenn der gleiche Startwert beim Aufruf benutzt wird, erzeugen die Funktionen immer die gleiche Folge von ``Zufallszahlen''. Zur Verschlüsselung von Daten ist ein Strom von zufälligen Bytes aus einer der Bibliotheksfunktionen deshalb nicht geeignet.

Linux bietet ein spezielles ``Gerät" zur Erzeugung kryptographisch hochwertiger Zufallszahlen an. Der Kernel verwaltet einen 512 Byte großen *Entropie-Pool*, der ständig aus der Beobachtung verschiedener unvorhersehbarer Ereignisse gespeist wird. Die produktivste Quelle dieser Zufallsereignisse ist die Tastatur: die letzte Stelle der mikrosekundengenauen Zeitmessung zwischen zwei Tastendrücken lässt sich weder voraussagen noch reproduzieren.

Über die Gerätedatei `/dev/random` kann auf den Inhalt des Entropie-Pools zugegriffen werden. Wenn alle 512 Bytes daraus entnommen wurden, liefert das ``Gerät" weitere Zufallsbytes erst, nachdem eine ausreichende Zahl zufälliger Ereignisse eingetreten ist.

Die Gerätedatei `/dev/urandom` hat diese Volumenbeschränkung nicht. Wenn der Entropie-Pool erschöpft ist, liefert dieses Device weitere Zufallszahlen auf der Basis des existierenden Pools. Diese Zahlen sind zwar noch statistisch gleichverteilt, es besteht aber eine gewisse innere Abhängigkeit der Zahlen untereinander. Für kryptographische Zwecke sind die Zufallszahlen aus dem zweiten Device deshalb nicht geeignet.

## **Die Senke `/dev/null`**

Diese Spezialdatei ist für alle Anwender zum Lesen und Schreiben frei. Sie ist der Mülleimer des Systems. Alles, was dorthin geleitet oder verschoben wird, verschwindet auf Nimmerwiedersehen.

## **Die Quelle `/dev/zero`**

Aus der Spezialdatei `/dev/zero` können beliebig viele Nullbytes gelesen werden. Diese Datei sollte nur mit Vorsicht als Quelle für Daten benutzt werden, weil die Menge der gelieferten Bytes nicht begrenzt ist.

## **Die Fehlerquelle `/dev/full`**

Beim Versuch, in die Datei `/dev/full` zu schreiben, wird der Fehler `ENOSPC` erzeugt, so als ob kein freier Platz mehr auf dem Speichermedium vorhanden wäre.

## **Der IO-Bereich `/dev/port`**

Über die Spezialdatei `/dev/port` können einzelne IO Ports angesprochen werden.

## **Der Arbeitsspeicher `/dev/mem` und `/dev/kmem`**

Die Datei `/dev/mem` bildet den physikalischen RAM an, die Datei `/dev/kmem` repräsentiert den virtuellen Speicher des Kernels. Beide Devices sind zeichenorientiert.

Auf diese Devices greifen nur spezielle Programme wie `free` oder `ps` zu. Das direkte Lesen und

Schreiben im Arbeitsspeicher des Rechners muß den Benutzern immer verboten sein.

# Runde Scheiben

Als Ergänzung und Erweiterung des Arbeitsspeichers haben alle PCs Massenspeicher, auf denen Daten nicht flüchtig bereitgehalten werden. Um einen direkten Zugriff auf diese Daten zu haben, werden sie in kleine Blöcke zerteilt (typischerweise 512 bis 2048 Bytes) und auf rotierenden Datenträgern gespeichert.

## Die Diskettenlaufwerke `/dev/fd*`

Die Dateien `/dev/fd*` repräsentieren die Diskettenlaufwerke (floppy disks). Linux kann ein oder zwei Floppycontroller mit jeweils zwei Diskettenlaufwerken, zusammen also maximal vier Laufwerke, verwalten.

Da die Diskettenlaufwerke in der Regel nicht zur alltäglichen Arbeit mit dem Computer benötigt werden, kann der Floppytreiber auch als Laufzeitmodul nur im Bedarfsfall geladen werden. Auf diese Weise wird der Kernel kleiner und der sonst vom Floppytreiber belegte Arbeitsspeicher bleibt normalerweise frei.

Der Linux-Kernel hat die erforderlichen Parameter zur Verarbeitung von 31 verschiedenen Diskettenformaten gespeichert. Davon sind 11 Formate für 5 1/4 Zoll Disketten, 5 Formate für ED 3,5 Zoll Disketten und die restlichen 15 Formate für normale 3,5 Zoll Disketten.

Name	Nr.	Sec	Hds	Track	Zoll	KByte	Floppy	LW
d360	1	9	2	40	5 1/4	360	DD	DD
h1200	2	15	2	80	5 1/4	1200	HD	HD
D360	3	9	1	40	3,5	360	DD	DD
D720	4	9	2	80	3,5	720	DD	DD
h360	5	9	2	40	5 1/4	360	DD	HD
h720	6	9	2	80	5 1/4	720	DD	HD
H1440	7	18	2	80	3,5	1440	HD	HD
E2880	8	36	2	80	3,5	2880	ED	ED
CompaQ	9	36	2	80	3,5	2880	ED	ED
h1440	10	18	2	80	5 1/4	1440	HD	HD
H1680	11	21	2	80	3,5	1680	HD	HD
h410	12	10	2	41	5 1/4	410	DD	HD
H820	13	10	2	82	3,5	820	DD	DD
h1476	14	18	2	82	5 1/4	1476	HD	HD
H1722	15	21	2	42	3,5	1722	HD	HD
h420	16	10	2	83	5 1/4	420	DD	HD
H830	17	10	2	83	3,5	830	DD	DD
h1494	18	18	2	83	5 1/4	1494	HD	HD
H1743	19	21	2	83	3,5	1743	HD	HD
h880	20	11	2	80	5 1/4	880	DD	DD
D1040	21	13	2	80	3,5	1040	DD	DD
D1120	22	14	2	80	3,5	1120	DD	DD
h1600	23	20	2	80	5 1/4	1600	HD	HD
H1760	24	22	2	80	3,5	1760	HD	HD
H1920	25	24	2	80	3,5	1920	HD	HD
E3200	26	40	2	80	3,5	3200	ED	ED
E3520	27	44	2	80	3,5	3510	ED	ED
E3840	28	48	2	80	3,5	3840	ED	ED
H1840	29	23	2	80	3,5	1840	HD	HD
D800	30	10	2	80	3,5	800	DD	DD
H1600	31	20	2	80	3,5	1600	HD	HD

**Tabelle:** Die 31 im Kernel gespeicherten Floppyformate

Jedes der in der Tabelle [2.2](#) aufgeführten Diskettenformate ist über eine Gerätedatei ansprechbar.

Hauptgerätenummer für alle Floppydevices ist 2. Die Untergerätenummer (minor device number) für ein bestimmtes Format kann durch die folgende Formel errechnet werden:

$$\text{Minor} = \text{Formatnummer} * 4 + 128 * \text{Controllernummer} + \text{Laufwerkseinheit}$$



Die Formatnummer steht in der zweiten Spalte der Tabelle, die Controllernummer ist 0 für den ersten Floppycontroller, 1 für den zweiten und die Laufwerkseinheit ist 0 für Laufwerk A: und 1 für Laufwerk

B:.

Die Gerätedateien mit der Formatnummer 0 (`/dev/fd0`, `/dev/fd1`...) sind nicht für ein spezielles Diskettenformat programmiert. Beim Öffnen dieser Dateien versucht der Kernel, das Diskettenformat automatisch zu erkennen.

Bei gleicher Bauart des Mediums unterscheiden sich die Formate vor allem in der Anzahl der Spuren und der Sektoren pro Spur. Da allen Diskettenformaten die gleiche Sektorgröße von 512 Byte zugrunde liegt, unterscheiden sich die Diskettenformate zwangsläufig in ihrer Datenkapazität.

Die gängigen Standardformate für Disketten enthalten 80 Spuren. Alle modernen Diskettenlaufwerke können aber problemlos 82 Spuren bearbeiten, manche sogar 83. Die Kapazität einer Diskette läßt sich also erhöhen, indem zwei oder drei zusätzliche Spuren formatiert werden.

Eine 3,5 Zoll Diskette wird üblicherweise mit 18 Sektoren pro Spur formatiert.  Ohne die einzelnen Bytes dichter zu packen, können bis zu 21 Sektoren auf eine Spur geschrieben werden, indem der Abstand zwischen den Sektoren verringert wird. Um noch weitere Sektoren hinzuzufügen, können sie in Gruppen  zusammengefaßt und gemeinsam verwaltet werden. Durch die eingesparten Verwaltungsdaten kann maximal das Äquivalent von 24 normalen Sektoren auf einer Spur untergebracht werden. Bei Sektorzahlen ab 21 sinkt die Schreib- Lesegeschwindigkeit, weil die Sektoren nicht in monotoner Folge hintereinander geschrieben werden können. Indem jeder zweite Sektor übersprungen wird (interleave), ist sichergestellt, daß die Daten eines gerade gelesenen Sektors verarbeitet sind, bevor der nächste Sektor beim Schreib-/Lesekopf ankommt.

Bei 5 1/4 Zoll Disketten ist die übliche Sektorenzahl 15, sie kann auf 18 beziehungsweise 20 Sektoren erhöht werden.

Beim Einlegen einer neuen Diskette kann die Anzahl der Spuren nicht in einer angemessenen Zeit ermittelt werden, deshalb werden Formate mit mehr als 80 Spuren nicht automatisch erkannt. Das gilt auch für die im Kernel gespeicherten Formate 12-18. Diese Formate können anstelle der unspezifischen Gerätedatei (`/dev/fd0`...) über die entsprechende spezielle Gerätedatei angesprochen werden.

Für die dem Kernel unbekannten Formate besteht die Möglichkeit, die notwendigen Parameter mit dem Programm `setfdprm` der Diskette entsprechend zu setzen. Bei Verwendung der `mttools` zum Lesen und Schreiben auf DOS-formatierten Disketten werden alle möglichen Diskettenformate anhand der im Dateisystem gespeicherten Informationen automatisch erkannt und gesetzt.

Die Diskettenlaufwerke sind normalerweise für alle Anwender beschreibbar, indem entweder die Gerätedateien selbst den Schreibzugriff erlauben, oder indem die schreibenden Programme (beispielsweise die `mttools`) das SUID oder SGID Bit gesetzt haben, so daß der Anwender zur Laufzeit des Programms die Privilegien des Dateieigentümers oder der Gruppe hat.

Das `mount`-Kommando ermöglicht der Systemverwalterin, durch den Eintrag der `user` Option für `mount` in der Datei `/etc/fstab` jedem Systembenutzer das Einbinden eigener Disketten in das allgemeine Dateisystem zu erlauben. Normalerweise ist es aus Gründen der Systemsicherheit nur der Superuserin erlaubt, das [Dateisystem zu verändern](#).

# Die IDE-Festplatten

Praktisch jeder PC ist mit einem IDE-Controller ausgerüstet, an dem die "normalen" Festplatten angeschlossen werden können. Die Dateien `/dev/hd*` repräsentieren diese Geräte. Außerdem werden durch die Einführung von Enhanced-IDE weitere Geräte an dieser Schnittstelle unterstützt. Aktuell sind das vor allem die ATAPI CD-ROMs, theoretisch gibt es auch eine Unterstützung von ATAPI Magnetbandlaufwerken.

Linux kann bis zu vier Controller mit insgesamt acht Geräten verwalten. Der Kernel erkennt automatisch die beiden ersten Controller und die daran angeschlossenen Geräte durch *Autoprobing*. Weitere Geräte an den beiden zusätzlichen Controllern können durch entsprechende Kommandozeilenargumente auf dem [Bootprompt](#) beim Kernel angemeldet werden. Wenn die Bustreiber der Hardware die gemeinsame Benutzung von Interrupts zulassen, können mehrere Controller über den gleichen Interrupt betrieben werden, lediglich die IO-Adressen müssen sich in jedem Fall unterscheiden.

Zusätzliche Informationen zum Betrieb mehrerer IDE-Controller unter Linux finden Sie bei den Kernel-Sourcen in der Datei `./linux/Documentation/ide.txt`.

Die "rohen" Geräte werden als `/dev/hda`, `/dev/hdb` usw. angesprochen. Bei Festplatten muß mit diesen Devices besonders vorsichtig umgegangen werden, sie sollten nur mit `fdisk` oder anderen speziell dafür vorgesehenen Programmen bearbeitet werden. Eine versehentliche Änderung des ersten Datenblockes auf diesem Device vernichtet die Partitionstabelle und macht alle Daten auf der Festplatte bis auf weiteres unbrauchbar.

Die einzelnen Partitionen der Festplatten werden über die Untergerätenummern ausgewählt. Die Gerätedateien mit der Erweiterung 1-4 bezeichnen die primären Partitionen, höhere Zahlen bezeichnen logische Partitionen.

Um den Systembenutzern den unberechtigten Zugriff auf die Daten im Dateisystem nicht über eine Hintertür zu ermöglichen, dürfen alle Festplattendevices nur mit Rootpermissions gelesen oder beschrieben werden. Den Usern steht dann nur der kontrollierte Zugriff über das Dateisystem offen.

Eine CD-ROM wird nicht in Partitionen unterteilt. Ein ATAPI-Laufwerk als erstes Gerät am zweiten Controller wird deshalb nur über die Gerätedatei `/dev/hdc` angesprochen. Wenn die CD ein ISO-9660 Dateisystem enthält, kann sie mit dieser Gerätedatei ins Dateisystem eingebunden werden.

## Die SCSI-Festplatten `/dev/sd*`

Die Dateien `/dev/sd*` bilden die SCSI Festplatten ab. Die Major Device Nummer für die SCSI Festplatten ist 8. Die einzelnen Festplatten werden in Schritten zu 16 Minor Device Nummern für die Partitionen angesprochen. Die Namensgebung entspricht dem System bei den anderen Festplatten: ein Buchstabe für die Festplatte und eine Zahl für die Partition.

## Altmetallverwertung: `/dev/xd*`

Der Linux-Kernel unterstützt auch 8-Bit XT-Festplattencontroller. Die Festplatten und Partitionen sind nach dem gleichen Schema benannt wie die AT-Bus Platten.



## Das Dateisystem im Dateisystem: /dev/loop?

Durch das Loop-Device ermöglicht es Linux, eine Datei wie eine Diskette oder Festplattenpartition zu mounten. So kann die innere Struktur der Datei wie ein zweites Dateisystem im Dateisystem bearbeitet werden. Beispielsweise kann ein Bootdiskimage, wie es auf den CD-ROMs aller Linux-Distributionen enthalten ist, über das Loop-Device gemountet und bearbeitet werden, ohne es vorher auf eine Diskette zu schreiben:

```
[01] # losetup /dev/loop1 /mnt/cdrom/disks/boot01
[02] # mount -r /dev/loop1 /mnt/floppy/
[03] # ls /mnt/floppy/
boot.b          initdisk.gz  map          message      scsi1        scsi2
[04] # umount /mnt/floppy
[05] # mount -r /mnt/cdrom/disks/boot02 /mnt/floppy -o loop
[06] # _
```

Im ersten Kommando wird ein freies Loop-Device mit der Datei verbunden, die gemountet werden soll. Danach kann die Datei über die Gerätedatei des Loop-Devices wie eine normale Diskette in das Dateisystem eingebunden werden. Im fünften Kommando sehen Sie, wie das verbesserte Systemprogramm mount automatisch eine Datei mit einem freien Loop-Device verbinden kann.

Solange sich die Datei auf der CD-ROM befindet, können Sie selbstverständlich nur zum Lesen darauf zugreifen.

Wenn der Kernel DES oder IDEA als Verschlüsselungsfunktionen unterstützt, kann mit dem Loop-Device auch ein Crypto-Filesystem im Kernelspace erzeugt werden.

Bitte beachten Sie, daß das der Treiber für das Loop-Device zwar in den Kernelsourcen enthalten ist, die Programme `losetup` und das spezielle `mount`, die zur Unterstützung des Gerätes notwendig sind, in vielen Distributionen jedoch nicht vorhanden sind.

## Zusammenlegung mehrerer Partitionen: /dev/md?

In der Kernelversion 2.0 ist der Treiber für Multiple Devices enthalten. Wenn dieser Treiber einkompiliert wird, erlaubt Linux die Zusammenlegung mehrerer Festplattenpartitionen auf drei verschiedene Weisen:

### linear

Im linearen Modus werden die physikalischen Partitionen einfach aneinandergehängt und logisch zu einer größeren Partition verbunden. Dieser Modus soll in späteren Versionen auch die nachträgliche Vergrößerung existierender Partitionen ermöglichen.

### Raid0

Mit Raid0 werden zwei Partitionen so miteinander verschmolzen, daß logisch zusammenhängende Datenblöcke auf zwei Partitionen und damit möglicherweise auf zwei unterschiedlichen Festplatten gespeichert werden. Partitionen, die in diesem Modus verbunden werden, lassen sich nachträglich nicht mehr verändern. Durch die Verteilung der Daten auf mehrere Geräte lassen sich deutliche Geschwindigkeitssteigerungen erzielen.

### Raid1



Mit Raid1 werden alle Daten auf unterschiedlichen Partitionen gespiegelt, um beim Auftreten eines Fehlers auf einer Partition die Daten mit Hilfe der Spiegelkopie rekonstruieren zu können. Dieser Modus wird noch nicht vollständig unterstützt, sie ist auch in jedem Fall viel langsamer als eine entsprechende Hardwarelösung.

Weitere Informationen und die Sourcen zu den Programmen, die zur Benutzung der Multiple Devices erforderlich sind, finden Sie per FTP auf `sweet-smoke.ufr-info-p7.ibp.fr` unter `public/Linux/md035.tar.gz`.

## Die CD-ROM-Laufwerke

Zu Beginn der Linux-Geschichte hat neben dem Internet die Diskette als Medium zur Verbreitung von Linux noch eine große Rolle gespielt. Heute hat die CDROM diese Funktion ganz übernommen. Weil die CD gleichzeitig Transport- und Speichermedium ist, ist sie in der Regel sogar dem Internet überlegen.

Anders als bei Floppys und Festplatten, bei denen sich einige wenige Geräteschnittstellen als Standard durchgesetzt haben, sind bei den CD-Laufwerken viele unterschiedliche Produkte mit herstellerspezifischen Geräteschnittstellen auf dem Markt. Linux unterstützt eine Vielzahl dieser Geräte. Vorausgesetzt, der entsprechende Treiber ist beim Übersetzen des Kernels aktiviert worden und die Unterstützung für das ISO9660-Dateisystem ist ebenfalls eingeschaltet, können Sie die Geräte über die jeweiligen Gerätedateien ansprechen.

In der Tabelle [2.3](#) finden Sie eine Aufstellung aller Gerätedateien zu den CD-ROM-Treibern.

Dateiname & Major & Minor & Beschreibung  
Gerätedateien für CD-ROM

<code>hdb</code>	3	1	ATAPI CD-ROM als zweites Gerät am ersten IDE-Controller
<code>hd{c,d}</code>	22	0,64	ATAPI CD-ROM als erstes oder zweites Gerät am zweiten IDE-Controller
<b><code>srn</code></b>	11	<i>n</i>	SCSI CD-ROMs
<code>sonycd</code>	15		Sony CDU31A und CDU33A Dieser Treiber muß <b>immer</b> durch ein Kommandozeilenargument auf dem Bootprompt initialisiert werden.
<code>gscd</code>	16		GoldStar
<code>optcd</code>	17		Optics Storage CD
<code>sjcd</code>	18		Sanyo CD
<code>mcdx</code>	20		Mitsumi (Extended, Multisession)
<code>mcd</code>	23		Mitsumi
<code>cdu535</code>	24		Sony CDU-535 und CDU-531

<b>sbpcdn</b>	25	0-3	SoundBlaster Pro, erster Controller, $0 \leq n \leq 3$  Dieser Treiber unterstützt auch Geräte von Matsushita, Kotobuki, Panasonic, CreativeLabs, Longshine und Teac. Die Typenbezeichnungen dieser Geräte sind CR-521, CR-522, CR-523, CR-562 und CR-563. <b>Achtung:</b> es gibt original Soundblaster-Karten mit integriertem IDE-Controller. CD-Laufwerke, die an diesen Controller (IO-Port unterhalb 0x200) angeschlossen werden, müssen über den ATAPI-Treiber bedient werden.
<b>sbpcdn</b>	26	0-3	SoundBlaster Pro, zweiter Controller, $4 \leq n \leq 7$
<b>sbpcdn</b>	27	0-3	SoundBlaster Pro, dritter Controller, $8 \leq n \leq 11$
<b>sbpcdn</b>	28	0-3	SoundBlaster Pro, vierter Controller, $12 \leq n \leq 15$
aztcd	29		Aztech/Orchid/Okano/Warnes
cm206cd	32		Phillips CM-206

Zu allen CD-Laufwerken können Sie Kommandozeilenargumente zur Konfiguration des Treibers auf dem [Bootprompt](#) angeben.

## Zeichenorientierte Ein- und Ausgabegeräte

Die meisten interaktiven Ein- und Ausgabeoperationen geschehen an zeichenorientierten Geräten. Der Prototyp eines solchen Gerätes ist die Tastatur, die mit jedem Tastendruck ein Zeichen an den Computer sendet. Auch der Bildschirm arbeitet zeichenorientiert: selbst wenn ein Text bereits vollständig in einer Datei vorliegt, wird er bei der Darstellung auf dem Bildschirm doch Zeichen für Zeichen an das Gerät (beispielsweise die Grafikkarte) gesendet.

## Die Console und virtuelle Terminals

Die Kombination von Tastatur und Bildschirm als Standardein- und -ausgabe wird als Terminal bezeichnet. In der Terminologie der alten Großrechenanlagen sind Terminals separate Geräte, die über eine serielle Leitung mit dem Zentralrechner verbunden sind. Ein spezielles Terminal dient dem Operator zur Steuerung des Zentralrechners, das ist die Systemconsole.

Die Arbeitsweise moderner Workstations unterscheidet sich von diesem Modell, trotzdem haben sich die Begriffe mit leichten Abwandlungen erhalten. Bildschirm und Tastatur des PC sind das moderne Terminal. Wegen Multiuser-Multitasking muß dieses eine Terminal gleichzeitig als Standard-Ein-Ausgabe für viele Programme dienen. Um die vielen Standardkanäle auseinanderhalten zu können, benutzt das Betriebssystem das Konzept der virtuellen Terminals. Jeder Prozeß kann vom Kernel ein eigenes virtuelles Terminal anfordern und damit seine eigene *Session* eröffnen. Das Betriebssystem bestimmt, welchem virtuellen Terminal die reale Tastatureingabe zugeordnet wird und wann die Ausgabe eines virtuellen Terminals auf dem echten Bildschirm erscheint.

Der einfache Textbildschirm, auf dem die Meldungen beim Booten des Systems erscheinen, entspricht der Console des Zentralrechners. Sobald das Betriebssystem in den Multiusermodus wechselt, wird

diese Console durch 8 oder mehr virtuelle Consolen überlagert. Tastatur und Bildschirm sind immer nur mit einer virtuellen Console verbunden. Zwischen den virtuellen Consolen kann durch die Tastenkombinationen ALT-F1 bis ALT-F8 umgeschaltet werden. Jede virtuelle Console wird durch eine eigene Gerätedatei im Dateisystem dargestellt.

TTY & DEV & VCS & DEV & VCSA & DEV & Beschreibung  
Die Virtuellen Consolen

console	4/0					reale Console
tty	5/0	vcs	7/0	vcsa	7/128	aktive virtuelle Console
tty1	4/1	vcs1	7/1	vcsa1	7/129	1. virtuelle Console (ALT-F1)
tty2	4/2	vcs2	7/2	vcsa2	7/130	2. virtuelle Console (ALT-F2)

Die Spalte TTY zeigt die eigentlichen Terminaldateien, VCS und VCSA sind zusätzliche Devices für den Zugriff auf den Bildschirm der virtuellen Consolen. Die Spalten DEV geben zu allen Devices die Major/Minor Gerätenummernpaare an.

Über die *Virtual Console Screens*, VCS, kann auf Text im aktuellen Bildschirmspeicher der virtuellen Consolen zugegriffen werden. Die VCSA-Devices liefern zusätzlich zum Text (Buchstaben) die Zeichenattribute und in den ersten 4 Bytes Information über Anzahl der Bildschirmzeilen und -spalten sowie die aktuelle Cursorposition. Diese Gerätedateien können einerseits als Schnittstelle für die Mausunterstützung auf den virtuellen Consolen dienen, andererseits können aus diesen Devices die Daten für Screendumps der jeweiligen Bildschirme bezogen werden.

## Pseudoterminals für die grafische Benutzeroberfläche

Das Konzept der virtuellen Terminals wird auch für die graphische Benutzeroberfläche von Linux, dem X Window System, angewendet. Die Prozesse, die mit diesem System arbeiten, können vom Betriebssystem ein Pseudoterminal anfordern und die Ein-Ausgabekanäle mit diesem Terminal verbinden.

Im Unterschied zur virtuellen Console wird das Pseudoterminal einer X-Anwendung nicht mit dem Gerätetreiber für Tastatur und Bildschirm verbunden, sondern mit dem X Window Server. Als Bindeglied gehört zu jedem Pseudoterminal ein Pseudo-TTY-Master, an den sich der X-Server an koppeln kann, um darüber mit dem Pseudoterminal zu kommunizieren.

Diese Master-Slave-Paare sind als Gerätedateien im Verzeichnis /dev eingetragen. Alle zeichenorientierten Geräte mit der Hauptgerätenummer 2 sind Master, die mit der Hauptgerätenummer 3 sind Slaves. Die Namen der Masterdateien sind ptyp0, ptyp1 bis ptyef. Die dazugehörigen Slaves haben die Namen tty0, tty1 bis ttyef. Die beiden signifikanten Zeichen der jeweils 256 Dateinamen setzen sich aus 16 Serien mit den Buchstaben pqrstuvwxyzabcde und den 16 Hexadezimalziffern 0123456789abcdef zusammen.

Bei älteren Kernelversionen gab es nur 64 Pseudoterminalpaare, die sich die Hauptgerätenummer 4 mit anderen Geräten teilen mußten. Um die Kompatibilität mit älteren Linux-Distributionen zu erhalten, werden diese Gerätedateien mit Untergerätenummern ab 128 als zusätzliche Einträge der ersten 64 Pseudoterminals bislang noch unterstützt.

# Die serielle Schnittstelle

Vom BIOS des PC werden vier serielle Schnittstellen unterstützt, COM1 bis COM4. Normalerweise ist ein Rechner mit zwei dieser Ports ausgerüstet. Für die dritte und vierte Schnittstelle werden von den Hardwareherstellern in der Regel als Standard die Interrupts 3 und 4 angeboten. Die gemeinsame Benutzung eines Interrupts durch zwei Schnittstellenbausteine ist aber aus technischen Gründen mit diesen Karten unmöglich. Der Betrieb von vier ``normalen" Schnittstellen ist deshalb nur dann zulässig, wenn die Hardware die Auswahl von zusätzlichen, freien Interrupts erlaubt. Wenn Sie Ihren Rechner mit zusätzlichen seriellen Schnittstellen ausstatten möchten, beispielsweise um einen Modempool zu betreiben, empfiehlt sich der Einsatz spezieller Multiportkarten. Eine Liste der von Linux unterstützten Modelle finden Sie im Serial-HOWTO.

DOS & In & Out & Minor & IO-Port & IRQ  
Die Gerätedateien für die seriellen Schnittstellen

COM1	ttys0	cua0	64	0x3f8	4
COM2	ttys1	cua1	65	0x2f8	3
COM3	ttys2	cua2	66	0x3e8	(4)
COM4	ttys3	cua3	67	0x2e8	(3)
	ttys63	cua63	127		

Zu jeder seriellen Schnittstelle gehören zwei Gerätedateien. Die eine, `/dev/ttys*`, **blockiert** einen Prozeß beim Versuch, das Device zu öffnen, solange, bis durch ein *Carrier-Detekt* an der Schnittstelle vom Modem oder Terminal das Bestehen einer Datenverbindung angezeigt wird. Die andere, `/dev/cua*` arbeitet „non-blocking“, also unabhängig vom Carrier-Detect, und erlaubt es dem Prozeß, selbst den Status der Schnittstelle abzufragen und entsprechend darauf zu reagieren. Das Betriebssystem verbietet das gleichzeitige Öffnen beider Gerätedateien für eine Schnittstelle. Während das Device `ttys*` bei geöffnetem `cua*` blockiert liefert das andere im umgekehrten Fall eine Fehlermeldung an den öffnenden Prozeß.

Ein auf der Modemleitung `/dev/ttys1` wartendes `getty` wird beispielsweise automatisch solange blockiert, bis das Modem die Carrier-Detect Leitung „hoch“ setzt. Wenn in diesem Moment die Datei `/dev/cua1` geöffnet ist (weil gerade ein ausgehender UUCP-Call das Modem belegt, der Carrier also zu dieser Verbindung gehört), wird das `getty` weiter blockiert. Erst wenn ein auf „Auto-Answer“ gesetztes Modem eine ankommende Modemverbindung registriert und von sich aus den Carrier-Detect anzeigt, wird die von `getty` angeforderte Gerätedatei `/dev/ttys1` geöffnet und der Prozeß aus seiner Blockade erlöst, um die Login-Prozedur für den ankommenden Anruf durchzuführen.

Die Blockierung des `open(2)` Systemaufrufs für die Devices `/dev/ttys?` kann durch den Modus `'O_NONBLOCK' ('O_NDELAY')` umgangen werden. Das so geöffnete Device verhält sich dann wie das entsprechende `/dev/cua?`.

# Die Busmäuse

Besonders bei Notebooks kommen Mäuse beziehungsweise Trackballs oder Touchpads zum Einsatz, die nicht über die normale serielle Schnittstelle sondern über einen speziellen Controller mit dem Rechner verbunden werden. Diese Eingabegeräte werden unter der Sammelbezeichnung ``Busmäuse" von Linux unterstützt.

## Datei & Minor & Beschreibung Die Busmäuse

logibm		Die Logitech-Busmäuse werden meistens mit einem runden, 9-poligen Stecker an eine Steckkarte angeschlossen.
psaux	1	Die PS/2-Mäuse sind eigentlich keine Busmäuse im engeren Sinne, vielmehr werden sie über den PS/2-Port des Tastaturcontrollers gesteuert. Die Zeigegeräte von Notebooks sind in der Regel von diesem Typ.
inportbm	2	Die Inport-Busmäuse (von Microsoft) haben normalerweise den gleichen Stecker wie die Logitech-Busmäuse.
atibm	3	Diese weitere Variante der Inport-Mäuse wird mit den kombinierten ATI-XL Grafik/Maus-Adaptern geliefert.

Für Interrupts für die echten Busmäuse können über Kommandozeilenargumente auf dem Bootprompt eingestellt werden.

## Der Wecker: `/dev/rtc`

Der Name ``*Real Time Clock*" ist wohl etwas hoch gegriffen. Es handelt sich bei diesem Gerät um die batteriegepufferte CMOS-Uhr, die den Rechner beim Einschalten mit dem aktuellen Datum versorgt. Diese Uhr kann einem Benutzerprozeß durch den Hardwareinterrupt 8 asynchron Signale schicken, und auf diese Weise zur Steuerung des Programms beitragen.

Bitte beachten Sie, daß diese Uhr meistens auf die internationale Normalzeit (UTC) eingestellt ist und von der lokalen Zonenzeit abweicht. Je nachdem, wiviel Zeit seit dem Einschalten des Rechners und dem damit verbundenen Setzen der Systemuhr vergangen ist, kann die Systemzeit von der Zeit der CMOS-Uhr zusätzlich abweichen.

Weitere Informationen und ein kleines Demoprogramm zur Erläuterung der Funktionen finden Sie bei den Kernelsourcen in der Datei `./linux/Documentation/rtc.txt`.

## Die parallele Druckerschnittstelle `/dev/lp?`

Die Gerätedateien `/dev/lp?` bilden die parallelen Schnittstellen für Drucker im Dateisystem ab. Linux unterstützt bis zu drei Druckerschnittstellen.

## Datei & Minor & IO-Port & Bemerkung

## Die Druckerschnittstellen

lp0		0x3bc	Dieser Druckerport ist ``die zweite Wahl" und wird von manchen Controllern nicht unterstützt.
lp1	1	0x378	Das ist der eigentliche Standardport, dem normalerweise der Interrupt 7 zugeordnet wird.
lp2	2	0x278	Wenn eine zweite Druckerschnittstelle vorhanden ist, wird sie meistens auf diesem IO-Port mit dem Interrupt 5 betrieben.

Bei vielen modernen Rechnern befindet sich ein paralleler Schnittstellenbaustein onboard. Die Parameter lassen sich dann meistens im BIOS einstellen (beziehungsweise dort nachlesen).

In der Standardeinstellung benutzt Linux keine Interruptsteuerung zum Drucken. Nach der vorgegebenen Methode unternimmt der Druckertreiber eine bestimmte Anzahl von Versuchen, seine Daten an den Drucker abzugeben. Wenn der Drucker nach Ablauf dieser Versuche nicht alle Daten übernehmen konnte, wartet der Treiber eine kurze Zeit und gibt den Prozessor für andere Programme frei. Nach dieser Zeit beginnt die Schleife mit den Versuchen von neuem, so lange, bis der Drucker alle Daten angenommen hat. Diese Methode wird als *Polling* bezeichnet.

Bei einem schnellen Rechner führt das Polling zu keinem spürbaren Leistungsverlust. Durch eine Interruptsteuerung kann aber die Rechenzeit, die der Druckertreiber mit dem Durchlauf der Versuchsschleifen verbringt, für nützlichere Aufgaben verwendet werden. Der Drucker löst das Hardwaresignal aus, wenn sein Puffer bereit ist, neue Daten aufzunehmen. Der Treiber schickt dem Drucker so viele Daten, bis dieser seinen Puffer aufgefüllt hat, danach ``legt er sich schlafen" und wartet, bis er vom Kernel nach dem nächsten Interrupt vom Drucker wieder aufgeweckt wird.

Um die Interruptsteuerung einzuschalten, muß die Systemverwalterin mit dem Programm `tunelp` dem Treiber den entsprechenden Interrupt mitteilen.

```
[01] # /usr/sbin/tunelp /dev/lp1
/dev/lp1 using polling
[02] # /usr/sbin/tunelp /dev/lp1 -i 7
/dev/lp1 using IRQ 7
[03] # _
```

Die Interruptsteuerung wird ausgeschaltet, indem als Interruptnummer 0 angegeben wird.

Die parallele Schnittstelle läßt sich nicht nur zum Anschluß von Druckern verwenden. Linux unterstützt noch zwei weitere Anwendungen des Parallelports:

### IP-Vernetzung mit PLIP

Mit einem *Lap-Link-Kabel* am Parallelport können zwei Rechner über eine Distanz von bis zu 15 Metern verbunden werden und mit den TCP/IP Protokollen vernetzt werden, wie über eine Ethernetverbindung.

### SCSI Ersatz für Iomega ZIP-Drive

Als interessante Alternative zu Floppystreamern und den alten Wechselplatten bieten die ZIP-Drives mit einer Kapazität von 96MB vielfältige Möglichkeiten zur Speicherung von Daten aller Art. Diese Geräte gibt es in einer Version für den Parallelport, der vom Linux-Treiber dann wie ein SCSI-Host angesprochen wird.

Beide Anwendungen sind nicht mit der ``normalen" Verwendung als Druckerschnittstelle vereinbar. Wenn Sie den Parallelport auf diese Weise einsetzen wollen und abwechselnd auch einen Drucker an

der gleichen Schnittstelle betreiben möchten, müssen Sie alle Treiber als Module anlegen und diese nach Bedarf in den Kernel einbinden. Wenn Sie zwei Parallelports zur Verfügung haben, können Sie zwei Treiber fest in den Kernel einbauen und sie durch Kommandozeilenargumente auf dem Bootprompt in passender Weise initialisieren.

(Beispiel: boot: linux lp=0x378,7 ppa=0x278)

## Die Soundkarte

In Verbindung mit einer Soundkarte bietet ein Linux-PC umfangreiche Multimedia-Qualitäten. Der Kernel unterstützt fast alle gängigen Fabrikate und es gibt Utilities zur Bedienung aller Grundfunktionen der Soundkarten.

### Dateiname & Minor & Beschreibung Gerätedateien für die Soundkarte

mixer		Die meisten Soundkarten enthalten eine Einheit zum Mischen der verschiedenen Eingabekanäle. Dieses "Mischpult" kann über die Gerätedatei <code>mixer</code> angesprochen und durch entsprechende IO-Controls die Lautstärke der verschiedenen Kanäle abgestimmt werden. Programme wie <code>xmiser</code> oder die verschiedenen CD-Player bieten ein komfortables Front-End zu dieser Komponente der Soundkarte an.
sequencer	1	Zum Abspielen von Sounds über den Synthesizer der Soundkarte oder über ein am MIDI-Port angeschlossenes Gerät dient die Datei <code>sequencer</code> . Die Befehle, die an diese Schnittstelle gesendet werden, kommen als <i>Events</i> in eine Warteschlange und werden zu dem im Event festgelegten Zeitpunkt an das Gerät ausgeliefert.
midi00	2	Zur direkten Kommunikation mit einem MIDI-Gerät ohne die Zeitsteuerung des Sequencers dient die Gerätedatei <code>midi00</code> .
dsp	3	Das <i>Digital Sampling Device</i> <code>dsp</code> ist die direkte Schnittstelle zum Analog/Digital-Wandler der Soundkarte. Es ist möglich, rohe Samples direkt aus der Gerätedatei zu lesen und darüber wiederzugeben. Bei der Grundeinstellung werden Samples mit einer Tiefe von 8-Bit und einer Rate von 8000Hz erzeugt und gespielt. Durch entsprechende Einstellung der Schnittstelle können Programme wie <code>vrec</code> und <code>vplay</code> Samples bis zu 16-Bit und 44kHz aufnehmen und wiedergeben.
audio	4	Mit der Gerätedatei <code>audio</code> bietet der Kernel einen Teil der Funktionalität des gleichnamigen Gerätes der Spark Workstation an. Der Einsatz ist ähnlich wie der des Digital Sampling Device, die Daten werden jedoch im $\mu$ -Law-Format codiert.
sndstat	6	Ähnlich wie die Dateien des Proc-Filesystems liefert die Gerätedatei <code>sndstat</code> Informationen über den Status des Soundtreibers. Sie können die Daten durch das Programm <code>cat</code> anzeigen lassen.
music	8	Diese Gerätedatei ist eine andere Art von Sequencer, speziell für die MIDI-Schnittstelle. Die Gerätedatei kann auch unter dem Namen <code>sequencer2</code> verzeichnet sein.

Sollte mehr als eine Soundkarte installiert sein, können weitere Gerätedateien für Sampler und MIDI unterstützt werden.

# Daten am laufenden Band

Als Medium zur Datensicherung bieten sich Magnetbänder wegen ihres besonders günstigen DM/Megabyte-Verhältnisses an. Mit Linux können die gängigen Bandlaufwerke für PC betrieben werden.

Durch die physikalische Anordnung der Daten auf dem Magnetband ist ein direkter Zugriff auf einzelne Datenblöcke nicht möglich. Die Magnetbänder können immer nur sequentiell vom Anfang her gelesen werden. Bandlaufwerke werden als zeichenorientierte Geräte betrieben. Die Gerätetreiber können das Band automatisch nach dem Schließen der Gerätedatei zurückspulen („Rewind On Close“).

Weil die Bandgeräte mit kontinuierlichen Datenströmen arbeiten, werden sie auch als Streamer bezeichnet.

## Die QIC-02-Streamer `/dev/rmt?` und `/dev/tape*`

Linux unterstützt im Standardkernel neben SCSI Bandlaufwerken mit dem QIC-Aufzeichnungsformat auch Laufwerke mit eigener Adapterkarte nach dem QIC-02 Standard (beispielsweise Archive SC400/SC402/SC499, Everex 811V/831V oder Wangtek 5150).

Die Dateien `/dev/rmt?` bilden diese Streamer im Dateisystem ab. Die QIC-02 Bandgeräte benutzen die Hauptgerätenummer 12. Durch die Untergerätenummer wird der Treiber für eine bestimmte Bandsorte eingestellt:

**0**  
automatische Erkennung

**2**  
QIC-11 (24 MB, 4 Spuren, 10 000 ftpi)

**4**  
QIC-24 (60 MB, 9 Spuren, 10 000 ftpi)

**6**  
QIC-120 (120 MB, 15 Spuren, 12 500 ftpi)

**8**  
QIC-150 (150 MB, 18 Spuren, 12 500 ftpi)

Einträge für QIC-300 und QIC-600 (Minor 10 und 12) sind im Sourcecode für den Treiber enthalten, aber als ungetestet kommentiert.

Die Geräte mit den hier aufgezählten Nummern arbeiten „No Rewind On Close“; wenn das niedrigste Bit 1 (die Gerätenummer also ungerade) ist, arbeitet der Streamer „Rewind On Close“. Die Gerätedateien mit ungeraden Minor Device Nummern werden vom MAKEDEV Script nicht automatisch erzeugt.

Wenn das 7. Bit der Minor Device Nummer gesetzt ist, gibt der Gerätetreiber zusätzliche Debugging-Information aus. Die vom MAKEDEV Script der aktuellen Distributionen erzeugte Datei `/dev/tape-d` ist ein Beispiel für diesen Fall. Ein Öffnen der Gerätedatei `/dev/tape-reset` (Minor 255) bewirkt einen Reset des Bandcontrollers und erforderlichenfalls ein Zurückspulen des Bandes.



## Die SCSI-Streamer /dev/st\* und /dev/nst\*

Tabelle 2.9 gibt einen Überblick über die Gerätedateien für SCSI-Streamer. Die Dateien `nst*` unterscheiden sich von den `st*` Devices dadurch, daß die `n*` Versionen beim Schließen der Datei das Band nicht zurückspulen (*No Rewind On Close*). Das Zählmuster für die Untergerätenummern (Minor) kann für alle Modi durch zusätzliche Laufwerke ergänzt werden.

Die Gerätedateien für SCSI-Bandlaufwerke

Datei	st0	st1	st0l	st1l	st0m	st0a	nst0	nst0l	nst0m	nst0a
Minor		1	32	33	64	96	128	160	192	224
Modus			1	1	2	3		1	2	3

Eine spezielle Adressierung des Gerätetreibers zur Bestimmung des Magnetbandtyps ist nicht notwendig, weil die SCSI Streamer normalerweise die Bandsorte selbst erkennen. In Bezug auf die Schnittstelle macht der Kernel auch keinen Unterschied zwischen QIC-02 und DAT Tapes. Spezielle Einstellungen des Treibers, beispielsweise zur Änderung der Blockgröße, werden durch das Systemprogramm `mt` vorgenommen.

Um die Organisation eines Systems mit vielen unterschiedlichen Bändern mit verschiedenen Parametern zu erleichtern, bietet der Treiber für jede der beiden Grundbetriebsarten drei zusätzliche Gerätedateien, für die jeweils ein anderer Modus geladen werden kann. Die Modi werden beim Booten oder beim Laden des Treibermoduls alle gleich initialisiert und müssen jeweils für eines der zusätzlichen Devices mit dem Programm `mt` eingestellt werden. Sie gelten dann gemeinsam für die die Geräte mit und ohne Zurückspulen. Zusätzliche Informationen finden Sie bei den Kernelsourcen in der Datei `./linux/drivers/scsi/README.st`.

## Der Floppystreamer (ftape)

Die Gerätedateien `/dev/rft0` und `/dev/nrft0` dienen als Schnittstelle zum Gerätetreiber des ersten Floppystreamers. Diese Geräte arbeiten mit formatierten Magnetbändern der Typen QIC-40/-80 an einer QIC-117 Schnittstelle. Sie werden wie ein drittes Diskettenlaufwerk an den Floppycontroller angeschlossen. In der Kernelversion 2.0 kann der Treiber sowohl als Modul als auch als fester Bestandteil des Kernels eingebunden werden. Wenn Sie an einem zweiten Controller zusätzliche Streamer angeschlossen haben, können diese Geräte über weitere Devices mit entsprechend höheren Untergerätenummern angesprochen werden. Häufig wird ein symbolischer Link von `/dev/ftape` auf `/dev/rft0` und von `/dev/nftape` auf `nrft0` angelegt.

Die Operationen auf `/dev/nrft0` führen nach ihrer Beendigung nicht zum Zurückspulen des Bandes (*No Rewind On Close*). Wenn Sie den Treiber als Modul durch den Kerneldämon laden lassen, dürfen Sie das Band auf keinen Fall länger als eine Minute in dieser Position lassen, weil sonst der Kerneldämon den Gerätetreiber entfernt und beim nächsten Zugriff auf das Band Daten verloren gehen können.

**Next:** [Die Konfigurationsdateien im Verzeichnis](#) **Up:** [Reise durch's Dateisystem](#) **Previous:** [Die Binärverzeichnisse](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

**Next:** [Home, Sweet Home](#) **Up:** [Reise durch's Dateisystem](#) **Previous:** [Die Gerätedateien im Verzeichnis](#)

## Subsections

- [/etc/adjtime](#)
  - [/etc/fdprm](#)
  - [/etc/fstab](#)
  - [/etc/gettydefs](#)
  - [/etc/group](#)
  - [/etc/hosts](#)
  - [/etc/inittab](#)
  - [/etc/issue](#)
  - [/etc/ld.so.conf](#)
  - [/etc/login.defs](#)
  - [/etc/magic](#)
  - [/etc/man.config](#)
  - [/etc/motd](#)
  - [/etc/nologin](#)
  - [/etc/passwd](#)
  - [/etc/printcap](#)
  - [/etc/profile](#)
  - [/etc/psdatabase](#)
  - [/etc/rc\\*](#)
  - [/etc/securetty](#)
  - [/etc/shells](#)
  - [/etc/syslogd.conf](#)
  - [/etc/termcap](#)
  - [Nichts über TCP/IP](#)
-

# Die Konfigurationsdateien im Verzeichnis /etc


Im Verzeichnis /etc befinden sich Konfigurationsdateien für das Basissystem. Fast alle Dateien dieses Verzeichnisses enthalten normalen Text und können/müssen von der Systemverwalterin mit einem normalen Editor bearbeitet werden, um die von diesen Konfigurationsdateien gesteuerten Programme an das lokale System anzupassen.

Ein Listing des /etc-Verzeichnisses sieht beispielsweise so aus:

```
$ ls -F /etc
X11/          gettydefs    inittab      passwd       services
adjtime       group        issue        printcap     shells
csh.login     host.conf    ld.so.conf   profile      skel/
disktab       hosts        lilo.conf    protocols    syslog.conf
exports       hosts.allow  magic        psdatabase   termcap
fdprm         hosts.deny   motd         rc.d/        ttytype
fstab         hosts.equiv  mtab         resolv.conf
ftputers      hosts.lpd    mtools       rpc
gateways      inetd.conf   networks     securetty
$ _
```

Aus Gründen der Systemsicherheit sollten „normale“ Benutzer die Dateien im Verzeichnis /etc nur lesen, aber nicht verändern können.

## /etc/adjtime

Die Datei adjtime enthält die Daten zur Korrektur der batteriegepufferten CMOS-Uhr. Die Daten werden vom Systemprogramm clock ausgewertet. Weitere Informationen finden Sie auf Seite .

## /etc/fdprm

In der Datei /etc/fdprm können die Parameter von Diskettenformaten zur Programmierung des Floppycontrollers mit setfdprm festgehalten werden.

Jede Zeile, die nicht mit einem # als Kommentar gekennzeichnet ist, beschreibt ein Diskettenformat. Der erste Eintrag ist eine beliebige Zeichenkette zur namentlichen Bezeichnung des Formates. Die darauffolgenden 9 Einträge sind die eigentlichen Formatparameter.

Die Werte für die Formatparameter lassen sich unmittelbar nach dem Formatieren mit dem Kommando getfdprm ermitteln.

#Name	size	sec/t	hds	trk	stre	gap	rate	spec1	fmt_gap
23.80.2	3680	23	2	80	0	0x1c	0x14	0xcf	0x04
23.82.1	3772	23	2	82	0	0x1c	0x10	0xcf	0x04

23.82.2      3680      23      2      82      0      0x1c      0x14      0xcf      0x04

Weil der Kernel bereits 31 vordefinierte Formate automatisch erkennen und benutzen kann, ist die manuelle Einstellung des Diskettenformates in der Regel überflüssig.

Im Kernel sind keine 2m-Formate definiert. Wenn eine in diesem Format erstellte Diskette mit dem `mount`-Kommando in das Dateisystem eingebunden werden soll, müssen nach dem Einlegen der entsprechenden Diskette vor dem ersten Zugriff die entsprechenden Parameter geladen werden. In dem Beispiel oben sind unter 23.80.2 die Parameter für eine 2m-formatierte 3,5 Zoll Diskette mit 23 Sektoren pro Spur und 80 Spuren angegeben.

## **/etc/fstab**

In der Datei `/etc/fstab` sind die dauerhaften Parameter für die Zusammensetzung des Dateisystems gespeichert. Hier legt die Systemverwalterin fest, welche Partitionen an welcher Stelle in das Dateisystem eingehängt werden. Die Datei wird von dem Kommando `mount -a` ausgewertet, wenn es bei der Systeminitialisierung von einem der [Initialisierungsscripts](#) ausgeführt wird.

Die Einträge in dieser Datei haben die Form:

*Device    Mountpunkt    Typ    Optionen    Dump    Check*

Eine genaue Beschreibung dieser Datei finden Sie ab Seite .

## **/etc/gettydefs**

Diese Datei wird von dem im System V Stil arbeitenden `getty_ps` ausgewertet. Hier werden die Einstellungen des Terminaltreibers für die verschiedenen Geräte festgelegt. Eine genaue Beschreibung finden Sie im Abschnitt über [getty\\_ps](#).


## **/etc/group**

In dieser Datei sind die Benutzergruppen und ihre Mitglieder festgehalten.

Die Datensätze der `group` Datei haben folgendes Format:

*Gruppenname:Paßwort:Gruppennummer :Mitgliederliste*

Weitere Informationen zur Verwaltung von Benutzergruppen finden Sie auf Seite .

Mit dem [newgrp-Kommando](#) kann jeder hier eingetragene Anwender die aktuelle Benutzergruppe wechseln. Wenn im Paßwort Feld ein verschlüsseltes Paßwort wie im entsprechenden Feld der `/etc/passwd` Datei eingetragen ist, können alle Systembenutzer, die das unverschlüsselte Paßwort kennen, in diese Gruppe wechseln. 

## **/etc/hosts**

Die Datei `/etc/hosts` gehört zu den Konfigurationsdateien des TCP/IP Netzwerkes. Hier werden die typischen vier Byte langen IP-Adressen den verbalen Namen der Netzwerkrechner fest zugeordnet. In Jugend des Internet mußten in dieser Datei alle Rechner des internationalen Netzes eingetragen sein, zu denen eventuell irgendwann einmal eine direkt benannte Verbindung aufgebaut werden könnte (das heißt eigentlich jeder Rechner des Internet).

Unter Linux wird die Auflösung eines Rechnernamens in seine IP-Adresse (und umgekehrt) durch den Domain Name Service mit dem **bind**-Paket erledigt. Um bei Einzelrechnern und kleinen lokalen Netzwerken den administrativen Aufwand eines Nameservers zu sparen, kann trotzdem weiterhin die `/etc/hosts` Datei benutzt werden. Dazu wird der als `resolver` bezeichnete Teil des Nameservers durch entsprechende Einträge in der Datei `/etc/host.conf` veranlaßt, zuerst in der `hosts` Datei nach einer passenden Auflösung für eine Adresse zu suchen. Die Funktionen des `resolver` sind in der C-Standardbibliothek enthalten, deshalb wird er automatisch von allen Programmen mit Netzwerkfähigkeit (z.B. `ftp`, `telnet`, `smail`, aber auch vom X-Server) benutzt.

Die Einträge in `/etc/hosts` sind einfacher Text und bestehen aus den IP-Adressen der Hostrechner am Anfang einer Zeile und den offiziellen oder inoffiziellen Namen dieser Rechner, jeweils durch Leerzeichen oder Tabulatoren getrennt.

```
127.0.0.1      localhost
193.98.158.33  atlantis.lunetix.de atlantis
193.98.158.1   soho.lunetix.de  soho
193.98.158.2   cicero.lunetix.de cicero
193.98.158.5   ovid.lunetix.de ovid
```

## **/etc/inittab**

Die Datei `/etc/inittab` wird vom `init` Programm benutzt, das die darin festgelegten Prozesse startet und so das Benutzersystem initialisiert. Eine genaue Beschreibung dieser Datei finden Sie im Abschnitt über die [Systeminitialisierung](#).

## **/etc/issue**

Die Datei `/etc/issue` wird von `getty` vor dem Login-Prompt ausgegeben. Der Text, oder besser das Bild, das mit dieser Datei angezeigt wird, ist so etwas wie die Fassade des Systems.

Nach der Installation eines neuen Systems wird in diesem Text-Bild meistens der Name der Linux-Distribution präsentiert. Die Distributoren wenden gerne einen Trick an, um den Austausch dieser Titelseite zu verhindern: In einem der Initialisierungsscripts (beispielsweise `/etc/rc.d/rc.local`) wird bei jedem Systemstart die `issue`-Datei mit der Datei des Distributors überschrieben. Wenn Sie Ihrem System eine eigene Fassade geben wollen, müssen Sie diesen Mechanismus abstellen oder an Ihre eigenen Bedürfnisse anpassen.

## **/etc/ld.so.conf**

In der Konfigurationsdatei `/etc/ld.so.conf` werden die Verzeichnisse angegeben, in denen das Systemprogramm `ldconfig` nach neuen Shared Libraries sucht.

`ldconfig` erzeugt die Datei `/etc/ld.so.cache` und die symbolischen Links auf die dynamischen Laufzeitbibliotheken, mit denen der Laufzeitlinker `ld.so` die Shared Libraries identifiziert.

Die beiden Standardverzeichnisse für die Laufzeitbibliotheken, `/lib` und `/usr/lib` müssen in `/etc/ld.so.conf` nicht extra angegeben werden. Zusätzliche Verzeichnisse, in denen Shared Libraries für den Laufzeitlinker typischerweise installiert werden, sind `/usr/X11R6/lib` und `/usr/i486-linuxaout/lib`.

Das Programm `ldconfig` sollte bei jedem Systemstart automatisch von einem der Initialisierungsskripts aufgerufen werden, damit stets die aktuellsten Versionen aller Laufzeitbibliotheken verwendet werden.

## **/etc/login.defs**

Die Datei `/etc/login.defs` gehört zum `login` Programm aus dem Shadow Paßwortsystem. Es liest daraus die Parameter, mit denen beispielsweise eingestellt wird, wie oft ein fehlgeschlagenes Login wiederholt werden darf und ob und wo die Meldungen über solche Fehlschläge festgehalten werden. Hier können auch einige Umgebungsvariablen für alle Prozesse festgelegt werden, die in der vom `login` erzeugten Prozeßfamilie gestartet werden. Diese Datei ist sehr ausführlich kommentiert.

## **/etc/magic**

Die Datenbank `/etc/magic` wird vom Kommando `file` benutzt, um den Typ einer Datei festzustellen. Es vergleicht die hier gespeicherten Merkmale mit der zu untersuchenden Datei und gibt bei Übereinstimmung den hier verzeichneten Dateityp aus.

Diese Datei befindet sich häufig nicht im Verzeichnis `/etc` sondern in `/usr/lib`.

## **/etc/man.config**

In der Datei `/etc/man.config` wird die Laufzeitkonfiguration des Online-Hilfesystems `man` festgelegt. Hier kann der `MANPATH` bestimmt werden, das sind die Verzeichnisse, in denen nach den Manualpages gesucht wird. Hier kann auch die Reihenfolge verändert werden, in der die verschiedenen Sektionen nach einer Manualpage durchsucht werden.

## **/etc/motd**

Die *Message Of The Day* wird jedem User automatisch am Ende der Login-Prozedur angezeigt, noch bevor die Shell den ersten Prompt ausgibt. Der Zweck dieser Datei ist, wie der Name schon sagt, in möglichst plakativer Form über eine aktuelle Neuigkeit von allgemeiner Wichtigkeit zu informieren.

## **/etc/nologin**

Die Datei `/etc/nologin` wird nur vom `login` Programm benutzt. Wenn diese Datei existiert, ist jedes „normale“ Einloggen im System unmöglich. Nur die Superuserin (`root`) kann sich trotzdem beim System anmelden. Wenn ein anderer Benutzer versucht, sich einzuloggen, wird der Inhalt der Datei `/etc/nologin` ausgegeben.

Es ist ratsam, in der Datei `/etc/rc` bei der Initialisierung des Systems mit dem Kommando ``rm -f /etc/nologin'` eine eventuell noch vorhandene Sperrung zu lösen.

## **/etc/passwd**

Die Datei `/etc/passwd` ist die Benutzerdatenbank des Systems. Hier werden die Namen, die Benutzernummern und das Heimatverzeichnis der Anwender gespeichert. Außerdem werden in der „normalen“ `passwd` Datei auch die verschlüsselten Paßwörter gespeichert.

Die Datensätze der Paßwortdatei bestehen aus:

*Benutzername:Paßwort: Benutzernummer:Gruppennummer: GCOS:Heimat:Shell*

Jeder Benutzer kann mit dem Systemkommando `passwd` sein Paßwort selbständig ändern. Er sollte das in regelmäßigen Abständen und zu besonderen Anlässen auch tun.

Ausführlichere Informationen zur Benutzerverwaltung finden Sie ab Seite .

## **/etc/printcap**

Die Datei `/etc/printcap` enthält eine stark formalisierte Beschreibung des oder der Drucker des Systems. Sie wird vom `lpd` Druckerdämon ausgewertet, der die Druckjobs im System verwaltet. Eine Beschreibung ist im Kapitel über [Dämonen](#) zu finden.

## **/etc/profile**

Die Datei `/etc/profile` wird von den Loginshells aller Benutzer gelesen und als Shellscript ausgeführt. Hier werden Grundeinstellungen der Shellumgebung für alle User vorgenommen. Wenn sie nicht vor dem Überschreiben geschützt werden (siehe beim Shellkommando `typeset`), können alle Einstellungen von einer benutzereigenen Initialisierungsdatei wieder geändert werden.

# Beispiel für `/etc/profile`

```
PATH="/bin:/usr/bin:/usr/X11R6/bin:/usr/local/bin"
PS1="[ \u@\h \W] \\\$ "
PS2="> "
umask 022
ulimit -c 0
```



## **/etc/psdatabase**


Die Datei `/etc/psdatabase` enthält kernelabhängige Daten für das Systemprogramm `ps` (genauer `procp`s). Anhand der in dieser Datei abgespeicherten Informationen kann `ps` in einem ausführlichen Listing der Prozeßtafel (mit der Option `-l`) den Namen der Kernelfunktion anzeigen, in der ein Prozeß gerade schläft.

Die `psdatabase` kann nicht mit einem normalen Editor bearbeitet werden. Sie wird durch das spezielle Programm `psupdate` aus der Kerneldatei `/usr/src/linux/vmlinux` erzeugt.

Das kernelabhängige `ps` benötigt ebenfalls eine derartige Datei. Das Format ist jedoch nicht kompatibel. `kmem-ps` legt seine `psdatabase` normalerweise im Verzeichnis `/var/run` ab.

## **/etc/rc\***

Die Dateien `/etc/rc*` sind die Systeminitialisierungsdateien. Sie werden vom Systemprogramm `init` beim Booten der Shell übergeben, die die darin enthaltenen Kommandozeilen ausführt.

Eine ausführliche Beschreibung der Systeminitialisierung durch `init` finden Sie ab Seite  in diesem Buch.

## **/etc/securetty**

In der Datei `/etc/securetty` werden die Ports (Terminals) angegeben, an denen sich die Superuserin (`root`) einloggen darf. Diese Datei wird vom `login` Programm gelesen und ausgewertet.

Wenn Sie das Shadow-Paßwortsystem installiert haben, kann diese Datei durch den Eintrag ```CONSOLE /etc/securetty" in login.defs aktiviert werden.`

## **/etc/shells**

In der Datei `/etc/shells` sind alle verfügbaren (zugelassenen und uneingeschränkten) Loginshells eingetragen.

Sie wird vom `chsh`-Kommando ausgewertet. Dem Anwender wird damit die Möglichkeit gegeben, die in dieser Datei zeilenweise aufgelisteten Programme als Loginshell in der Datei `/etc/passwd` einzutragen.

Außerdem wird sie vom `ftp` Dämon benutzt, um festzustellen, ob ein Benutzer einen uneingeschränkten Shellaccount hat, um ihm in diesem Fall auch den uneingeschränkten FTP-Zugang zu gestatten.

## **/etc/syslogd.conf**

Der `syslogd` Systemschreiber erfährt aus dieser Datei, welche Meldungen er wohin schreiben soll. Eine genaue Beschreibung der Datei finden Sie [hier](#).

## **/etc/termcap**

Die Datei `/etc/termcap` ist eine Datenbank, in der die Steuersequenzen für verschiedene Terminals abgespeichert sind. Diese Datenbank ist stark formalisiert, kann aber mit jedem Editor gelesen und bearbeitet werden.

Viele Programme benutzen diese Datenbank, indem sie die Steuerzeichen für die Bildschirmausgabe und die ankommenden Tastaturcodes mit den entsprechenden Einträgen in der Datenbank übersetzen. Die zu einem Terminal passende Übersetzungstabelle wird in der `TERM` Umgebungsvariablen bestimmt, die vom `getty` Programm oder bei der Shellinitialisierung gesetzt wird.

Eine genaue Erklärung aller möglichen Einträge dieser Datei würde den Rahmen dieses Buches sprengen. Es gibt eine TEX info-Datei zu `termcap` und Tim O`Reilly hat Bücher über `termcap` und über die `curses`-Library herausgebracht, die Ihnen alle Fragen zu diesem Thema beantworten können.

## **Nichts über TCP/IP**

Das komplette Thema des TCP/IP Networking ist auch in dieser Auflage des Linux-Anwenderhandbuches ausgeklammert. Damit Sie einen groben Überblick erhalten, welche Dateien und Programme in diesen Bereich fallen, sind sie hier aufgeführt.

### **HOSTNAME**

Fully Qualified Domain Name

### **diphosts**

eine Art `/etc/passwd` für SLIP Login

### **exports**

Verzeichnisse und Rechnernamen, die diese Verzeichnisse per NFS mounten dürfen.

### **ftpaccess**

Woher? Wann? Was? ...

### **ftpusers**

... und Wer NICHT?

### **gateways**

Hier werden die Rechner des lokalen Netzes aufgeführt, die Kontakt mit andern Netzen haben.

### **host.conf**

eine der Konfigurationsdateien für den Domain Name Service

### **hosts.allow**

Hier werden die Rechner und Netze eingetragen, mit denen Verbindungen aufgebaut werden dürfen.

### **hosts.deny**

Die in dieser Datei aufgeführten Rechner und Netze können sich nicht mit diesem Rechner verbinden.

### **hosts.equiv**

Diese Rechner werden „gleichrangig“ behandelt.

## **inetd.conf**

In dieser Konfigurationsdatei wird bestimmt, welche Services vom `inetd` Netzwerkdämon gestartet werden.

## **named.boot**

eine Initialisierungsdatei für den `named` Dämon des Domain Name Services.

## **networks**

die Namen und IP-Adressen der lokalen Netze.

## **nntpserver**

der Name und die IP-Adresse des Newsrechners.

## **protocols**

die Liste aller Internetprotokolle, die vom Kernel unterstützt werden.

## **resolv.conf**

Diese Konfigurationsdatei bestimmt, ob und wo ein Nameserver erreicht werden kann.

## **rpc**

enthält die Zuordnung der durch den `rpc`-Service bereitgestellten Netzdienste zu Socketnummern.

## **services**

Diese Datei enthält eine Liste aller „well known services“ sowie einer Reihe weiterer Standardservices im Internet.

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [Home, Sweet Home](#) **Up:** [Reise durch's Dateisystem](#) **Previous:** [Die Gerätedateien im Verzeichnis](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

**Next:** [Die Bibliotheken in ./lib](#) **Up:** [Reise durch's Dateisystem](#) **Previous:** [Die Konfigurationsdateien im Verzeichnis](#)

## Subsections


- [Das möblierte Zimmer](#)

---

# Home, Sweet Home

Nachdem die Reise in den beiden vorhergehenden Verzeichnissen durch Neuland und unwegsames Gelände führte, erreichen Sie nun in einem Unterverzeichnis von /home Ihren persönlichen Bereich im Linux-System.

Im /home Verzeichnis befinden sich die privaten Verzeichnisse, die Heimatverzeichnisse aller Systembenutzer. Die Existenz solcher individuellen Bereiche ist für ein Mehrbenutzersystem selbstverständliche Notwendigkeit. Auch wenn Sie die einzige Benutzerin Ihres Systems sind, sollten Sie sich einen Account ohne Privilegien und ein Heimatverzeichnis anlegen.

In Ihrem Heimatverzeichnis können Sie nach Belieben Verzeichnisse anlegen und Daten speichern und löschen.  Hier werden Sie Ihre persönliche eMail aufbewahren und Ihre Projektdaten anlegen, zum Beispiel die Entwürfe für ein neues, besseres Linux-Handbuch.

Im Sinne des File-System-Standard zählt der /home Bereich damit eindeutig zum variablen, unbegrenzt wachsenden Systembereich. In lokalen TCP/IP Netzwerken, in denen Sie als Benutzerin an verschiedenen Rechnern arbeiten können, wird das /home Verzeichnis möglicherweise über NFS verteilt auf allen Rechnern des Netzes gemeinsam benutzt. Das hat den Vorteil, daß Sie an jedem Arbeitsplatz immer in dem selben Heimatverzeichnis arbeiten.

Auch wenn der /home Bereich lokal angelegt ist, wird er normalerweise auf einer eigenen Partition untergebracht. Diese physikalische Trennung der System- und Benutzerdaten erhöht die Betriebssicherheit, sie erleichtert ein Systemupdate und die Datensicherung.

## Das möblierte Zimmer

Wenn Sie sich an einem Linux-Rechner einloggen, ist Ihr Heimatverzeichnis immer das erste Arbeitsverzeichnis. Neben Ihren privaten Daten befinden sich deshalb in diesem Verzeichnis noch individuelle Initialisierungsdateien, mit denen Sie Ihre Arbeitsumgebung einrichten und in dem für eine Computeroberfläche möglichen Rahmen individuell gestalten können.

Eine Standardausstattung mit Initialisierungsdateien wird Ihnen von der Systemverwalterin beim Einrichten Ihres Accounts ins Heimatverzeichnis gestellt. Bei einem normalen Listing werden Sie diese Dateien nicht sehen, weil ihre Namen mit einem Punkt beginnen. Durch den Schalter **-a** (für all) werden aber auch die „versteckten“ Dateien in ihrem Heimatverzeichnis angezeigt.

```
$ ls -F ~
Mail/      News/      bin/       public/
$ ls -aF ~
```

./	.bash_login	.seyon/	Mail/
../	.bash_logout	.tcshrc	News/
.Xmodmap	.bashrc	.term/	bin/
.bash_history	.emacs	.xinitrc	public/

\$ \_

Die „sichtbaren“ Verzeichnisse Mail und News gehören zum Mailer-Frontend bzw. zum Newsreader und werden automatisch erzeugt, wenn sie beim ersten Aufruf des entsprechenden Programms nicht existieren. ~/bin kann in den Suchpfad der Shell integriert werden und ist für private, ausführbare Dateien vorgesehen. public dient zur Aufbewahrung von Dateien, die auch anderen Systembenutzern zugänglich sein sollen.

Die „unsichtbaren“ Dateien .Xmodmap und .xinitrc werden im Kapitel zum X11 Window System erklärt. Die Verzeichnisse .seyon und term werden im Kapitel über Datenreisen beschrieben.

Alle Dateien, deren Namen mit .bash beginnen, gehören zur bash, der Bourne-Again-Shell von Chet Ramey und Brian Fox. Die Funktion dieser Dateien ist mit Beispielen in der Kommandobeschreibung [hier](#) erklärt.

Die verbleibenden Dateien .tcshrc und .emacs sind an keiner anderen Stelle des Buches erklärt, deshalb folgt hier jeweils ein Beispiel.

```
$ cat .tcshrc
# tcshrc
if ($?prompt) then
    umask 022
    set cdpath = ( ~ /usr/src /usr )
    set notify
    set history = 100
    setenv OPENWINHOME /usr/openwin
    set path = ( /bin /usr/bin /usr/local/bin /usr/X11R6/bin\
        $OPENWINHOME/bin /usr/games ~/bin . )
endif
set prompt = "%m:%~%# "
set term = console
alias pwd 'echo $cwd'
alias ls 'ls -F'
alias term 'term < /dev/cua1 > /dev/cua1 2> /dev/null &'
$ _
```

```
$ cat .emacs
(add-hook 'text-mode-hook '(lambda () (auto-fill-mode 1)))
(standard-display-european 1)
(transient-mark-mode 1)
$ _
```

**Next:** [Die Bibliotheken in ./lib](#) **Up:** [Reise durch's Dateisystem](#) **Previous:** [Die Konfigurationsdateien im Verzeichnis](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

## Subsections

- [Compiler, Bibliotheken und Daten in /usr/lib](#)
    - [Der C-Compiler](#)
- 


# Die Bibliotheken in `./lib`

Im Verzeichnis `/lib` befinden sich die „Shared Libraries“ für die dynamisch gelinkten Programme des Basissystems. Weitere Shared Libraries für Zusatzprogramme und das X Window System sind in `/usr/lib` und `/usr/X11R6/lib`.

Normalerweise haben die Shared Libraries Namen der Form `libxyz.so.1.2.33`. Die dynamischen Linker öffnen die Shared Libraries über symbolische Links der Form `libxyz.so.1`. Wenn der symbolische Link immer auf die neueste Bibliotheksversion zeigt, wird der Linker veranlaßt, die Programme stets mit der aktuellsten Version zu verbinden. Das Systemprogramm `ldconfig` sorgt dafür, daß die symbolischen Links stets auf die Bibliotheken mit der höchsten Versionsnummer zeigen.

In `/lib` sind auch die dynamischen Linker, `ld.so` und `ld-linux.so` zu finden, der die dynamisch gelinkten Programme im aout- und ELF-Format beim Laden (zur Laufzeit) mit den Bibliotheksfunktionen zusammenbindet.

Um die Rückwärtskompatibilität zu sichern, wird ein symbolischer Link von `/lib/cpp` auf den C-Präprozessor zugelassen.

Das Verzeichnis `/lib/modules` enthält ein Unterverzeichnis für jede Kernelversion, in dem die zu dieser Version passenden Kernelmodule gespeichert werden. Weitere Informationen zu Kernelmodulen und ihrer Benutzung finden Sie ab Seite .

## Compiler, Bibliotheken und Daten in `/usr/lib`

Der Anspruch dieses Kapitels -- den Inhalt der verschiedenen Verzeichnisse im Basissystem zu erklären -- ist im Fall von `/usr/lib` nur ansatzweise zu erfüllen. In diesem Verzeichnis befinden sich alle möglichen Daten, die von Programmen des Basissystems zur Laufzeit benötigt werden.

Unter anderem sind das Hilfstexte, Makros für verschiedene Textverarbeitungssysteme, Funktionsbibliotheken für verschiedene Compiler und Interpreter, Fontdateien und alles, was sonst keinen Platz im Dateisystem gefunden hat.

## Der C-Compiler

Die größte Zahl der Dateien in `/usr/lib` gehört zum GCC, dem C-Compiler. Folgende Dateitypen lassen sich dem C-Entwicklungssystem zuordnen:

**`lib*.a`**

Statische Funktionsbibliotheken -- in diesen Dateien sind die Standardfunktionen enthalten, die zum Linken statischer Programme benötigt werden.

#### **lib\*.so.\***

Dynamische Funktionsbibliotheken -- in diesen Dateien sind die gleichen Funktionen enthalten, die in den gleichnamigen statischen Bibliotheken zu finden sind, allerdings werden diese Funktionen erst zur Laufzeit durch den dynamischen Linker mit der Programmdatei verbunden.

#### **cr\*.o**

Initialisierungsobjekte -- diese Objektdateien enthalten den Programmtext zur Initialisierung von Programmen oder Klassen.

#### **gcc-lib**

Verzeichnis -- in den Unterverzeichnissen befinden sich die eigentlichen Compiler, Präprozessoren und die compilerabhängigen Dateien des GCC.

Durch die Einführung eines neuen Binärformats für Linux ist das Compilersystem nicht gerade übersichtlicher geworden. Für das alte aout-Format werden andere Bibliotheken benötigt als für das neue ELF. Wenn vom Entwicklungssystem beide Formate unterstützt werden, befinden sich Bibliotheken und Binutils für das aout-Format in /usr/i486-linuxaout/\*. Zu den Shared Libraries von aout gibt es noch jeweils eine *Stub*-Library mit den Programmtexten für die Funktionsaufrufe.

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [Erweiterungspakete in /opt](#) **Up:** [Reise durch's Dateisystem](#) **Previous:** [Home, Sweet Home](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)



# Erweiterungspakete in /opt

Für kommerzielle Erweiterungssoftware, die nicht Bestandteil der Linux-Distribution ist, bietet der Entwurf des Filesystem-Hierarchie-Standard einen zusätzlichen Zweig des Verzeichnisbaums auf /opt an. Hier kann jedes zusätzliche Softwarepaket ein eigenes Unterverzeichnis anlegen und seine statischen Daten installieren.

**Next:** [Die temporären Dateien im](#) **Up:** [Reise durch's Dateisystem](#) **Previous:** [Erweiterungspakete in /opt](#)

# Die Prozeßdaten im Verzeichnis `/proc`

Das Verzeichnis `/proc` beherbergt das Prozeßdateisystem. Dieses Pseudodateisystem stellt viele interessante und wichtige Kernelinformationen in der Form von Dateien dar, die direkt (z.B. mit dem Programm `cat`) oder von speziellen Front-Ends aufbereitet angezeigt werden können.

Mehr Information zu diesem Thema finden Sie im Kapitel über Dateisysteme, [hier](#).

# Die temporären Dateien im Verzeichnis

## `./tmp`

Im Verzeichnis `/tmp` werden von einigen Programmen temporäre Dateien zur Zwischenspeicherung von Laufzeitdaten angelegt. Zum Erzeugen einer Datei in diesem Verzeichnis braucht ein Prozeß keine besonderen Rechte. Das Löschen von Dateien ist nur den Prozessen des Dateieigentümers erlaubt, wenn das Stickybit für das `/tmp` Verzeichnis gesetzt ist.

Weil in `/tmp` Dateien unvorhersehbarer Größe zu unberechenbaren Zeitpunkten erzeugt werden und um die übermäßige Fragmentierung der Rootpartition zu verhindern, wird das Verzeichnis `/tmp` normalerweise nicht im Rootfilesystem angesiedelt. Zur „Umsiedlung“ gibt es verschiedene Möglichkeiten: zum Beispiel kann anstelle eines echten Verzeichnisses ein symbolischer Link auf das Verzeichnis `/var/tmp` angelegt werden. Eine andere Möglichkeit besteht darin, eine eigene Partition oder auch eine RAM-Disk auf dieses Verzeichnis aufzusetzen.

Der temporäre Charakter der Daten im `/tmp`-Verzeichnis wird dadurch unterstrichen, daß der Inhalt dieses Verzeichnisses in beliebigen Abständen, beispielsweise bei jedem Systemstart, gelöscht wird.

Das Verzeichnis `/usr/tmp` existiert zur Kompatibilität mit traditionsbewußten, alten Programmen und ist in der Regel ein Link auf `/var/tmp`. Letzteres ist das auf einer Festplatte beheimatete Standardverzeichnis für alle temporären Daten des lokalen Systems.

**Next:** [Das Verzeichnis /var](#)
**Up:** [Reise durch's Dateisystem](#)
**Previous:** [Die temporären Dateien im](#)

## Subsections

- [/usr/X11R6](#)
  - [/usr/dict](#)
  - [/usr/doc](#)
  - [/usr/games](#)
  - [/usr/include](#)
  - [/usr/local](#)
  - [/usr/info](#)
  - [/usr/man](#)
  - [/usr/share](#)
  - [/usr/src](#)
  - [/usr/spool](#)
- 

# Das Verzeichnis /usr

Während die anderen Verzeichnisse der ersten Hierarchie für den „normalen“ Anwender wenig Interessantes zu bieten haben, eröffnet sich im /usr Verzeichnis die ganze Welt der Linux-Anwendungen. Hier ist der X11-Server ebenso zu finden wie das TEX Satzsystem, der GNU C-Compiler, der emacs Editor oder das News&Mail System.

Alle Programme des /usr Systems sind für alle Rechner eines lokalen Netzes gleichermaßen interessant und/oder wichtig. Deshalb wird mit dem File-System-Standard festgelegt, daß die Daten im /usr Ast des Dateisystems über ein lokales Netz verteilt benutzbar sein sollen. Eine wichtige Voraussetzung dafür ist, daß dieser Ast „Read-Only“ gemountet werden kann.

Um das zu erreichen, wird ein weiterer Ast des Dateisystems mit dem Namen /var eingeführt, der die variablen Teile des /usr Systems aufnimmt.

Ein beispielhaftes Listing des /usr-Verzeichnisses zeigt die folgenden Unterverzeichnisse:

```
$ ls -F /usr
X11R6/      dict/      games/     lib/       preserve@  spool@
adm@        doc/       include/   local/     sbin/      src/
bin/        etc/       info/      man/       share/     tmp@
$ _
```

Die Einträge adm, preserve, spool und tmp sind symbolische Links auf entsprechende Verzeichnisse unter /var.

## **/usr/X11R6**

In den Unterverzeichnissen von `/usr/X11R6` befindet sich das X Window System - die grafische Benutzeroberfläche von Linux.

## **/usr/dict**

Dieses Verzeichnis ist für die Wörterbücher (dictionaries) der `ispell` Rechtschreibkorrektur vorgesehen.

## **/usr/doc**

Der File-System-Standard sieht dieses Verzeichnis für verschiedene Dokumentationen von allgemeinem Interesse vor.

## **/usr/games**

```
Would you like to play a game? ttt
Funny, the only way to win is not to play at all.
```

## **/usr/include**

Das Verzeichnis `/usr/include` mit seinen Unterverzeichnissen enthält ausschließlich 'header' Dateien, die vom C-Präprozessor bearbeitet werden.

Das Verzeichnis `/usr/include/linux` sollte ein symbolischer Link auf das Verzeichnis `/usr/src/linux/include/linux` sein; `/usr/include/asm` sollte in gleicher Weise auf das Verzeichnis `/usr/src/linux/include/asm` zeigen.

## **/usr/local**

Um eine deutliche Trennung zwischen der originalen Distribution und den lokalen Erweiterungen zu erreichen, gibt es hinter dem Verzeichnis `/usr/local` eine Hierarchie von Verzeichnissen, die im Prinzip der von `/usr` entspricht. Der File-System-Standard legt fest, daß keine Linux-Distribution Daten in diesem Zweig des Verzeichnisbaums anlegen darf.

## **/usr/info**

In diesem Verzeichnis sind die Textdateien des T<sub>E</sub>Xinfo Dokumentationssystems untergebracht.

## **/usr/man**

Hier sind die Hilfstexte zu allen Kommandos, Dateiformaten, Spielen, C-Bibliotheksfunktionen etc., die sogenannten Manualpages, untergebracht.

Um landessprachliche Hilfstexte für alle Nationalitäten gleichberechtigt zu unterstützen, sieht der File-System-Standard eine Aufteilung von `/usr/man` in sprachspezifische Unterverzeichnisse vor.

Die zu den meisten Programmen gehörenden englischen Manualpages gehören nach `/usr/man/en`, die bereits existierenden oder noch entstehenden deutschen Hilfstexte werden im Verzeichnis

/usr/man/de\_DE.88591 untergebracht.

Die sprachspezifischen Verzeichnisse werden in der traditionellen Weise weiter unterteilt. Es sind zwei Gruppen von Unterverzeichnissen möglich. Die Namen der ersten und in jedem Fall erforderlichen Gruppe beginnen mit „man“ und enthalten die „rohen“ Manualpages. Diese unformatierten Hilfstexte werden von dem Programm man automatisch für die Anzeige auf dem Textbildschirm formatiert und durch einen Pager ausgegeben. Die rohen Manualpages können aber auch durch groff für andere Ausgabegeräte, beispielsweise einen Postscriptdrucker oder ein X Fenster, formatiert werden.

Die Namen der anderen Gruppe beginnen mit „cat“ und enthalten die Manpages bereits für den Textbildschirm formatiert. Die dort abgelegten Dateien können mit compress oder gzip komprimiert sein. Das man-Kommando dekomprimiert den Inhalt automatisch vor der Ausgabe.

Die Verzeichnisse jeder Gruppe sind numeriert. Die Numerierung entspricht der in der Erklärung vom man-Kommando beschriebenen [Aufteilung der Manpages](#).

## **/usr/share**

Die Funktion des Verzeichnisses /usr/share besteht in der Zusammenfassung architekturunabhängiger Daten in verteilten Systemen mit unterschiedlichen Rechnerarchitekturen.

## **/usr/src**

Dieser Zweig des Dateisystems ist für die Aufnahme der Quelltexte (Sourcen) für alle Programme des Standardsystems vorgesehen.

Eine sinnvolle Unterteilung in Unterverzeichnisse bleibt der Systemverwalterin überlassen. Allein das Verzeichnis /usr/src/linux mit den Kernelsourcen wird von anderen Systemkomponenten genau an dieser Stelle erwartet. Besonders wichtig sind die symbolischen Links der Verzeichnisse /usr/include/linux und /usr/include/asm auf die entsprechenden Verzeichnisse in /usr/src/linux/include.

## **/usr/spool**

Das Verzeichnis /usr/spool wird aus Gründen der Abwärtskompatibilität als symbolischer Link auf das neue Verzeichnis /var/spool weitergeführt.

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [Das Verzeichnis /var](#) **Up:** [Reise durch's Dateisystem](#) **Previous:** [Die temporären Dateien im](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

# Das Verzeichnis /var

Die Festlegung des /usr Verzeichnisses auf statische Daten mit dem Ziel, dieses Verzeichnis über NFS Read-Only mounten zu können, erfordert zwangsläufig die Auslagerung aller Dateien und Verzeichnisse, die für eine zweckmäßige Nutzung während des Betriebs verändert werden können/müssen. Diese Daten befinden sich dem File-System-Standard entsprechend in den Unterverzeichnissen von /var.

Ein Listing zeigt beispielsweise folgendes Bild:

```
$ ls -F
account/    cron/       local/      mail/       run/        tmp/
cache/      games/      lock/       opt/        spool/      yp/
crash/      lib/        log/        preserve/   state/
$ _
```

Im Verzeichnis /var/cache befinden sich Unterverzeichnisse, in denen verschiedene Programme redundante Daten ablegen können, die aus Gründen der Performance in halbverarbeiteter Form zwischengelagert werden sollen. Beispielsweise können hier vorformatierte Manualpages in /var/cache/man oder Fontdateien für L<sup>A</sup>T<sub>E</sub>X in /var/cache/fonts abgelegt werden.

Das Verzeichnis /var/log enthält die Systemlogfiles. Insbesondere wird hier das "Gedächtnis" aller Logins, die Datei wtmp und die Logfiles vom syslogd angelegt.

/var/lock ist das vom File-System-Standard empfohlene Verzeichnis zum Anlegen der Lockfiles.

/var/preserve ist das Sicherungsverzeichnis, in dem elvis nach einem Crash das unfertige Textfile automatisch sichern kann.

In /var/run werden Dateien mit Informationen zum laufenden System angelegt. Insbesondere ist das die Datei utmp, in der alle aktuell eingeloggten User verzeichnet sind. Alle Dateien aus /var/run sollten zu Beginn der Systeminitialisierung gelöscht werden, damit sich die hier enthaltenen Daten tatsächlich alle auf diesen Lauf beziehen.

In Verzeichnis /var/spool werden die Daten der einzelnen News&Mail Programme in entsprechende Unterverzeichnisse abgelegt. Je nach Auslegung des Systems kann hinter dem Verzeichnis /var/spool/news eine sehr große Hierarchie beginnen, die für jede Nachrichtenrubrik ein Verzeichnis enthält. In diesen Verzeichnissen sind dann die einzelnen Artikel in nummerierten Dateien abgelegt. Im Verzeichnis /var/spool/uucp werden die für eine Netzwerkkopie mit dem uucp Programm bzw. mit dem uucico Daemon vorgesehenen Dateien zwischengelagert. In /var/spool/lp werden die Dokumente des lpd Druckerspoolers zwischengespeichert, wenn in /etc/printcap kein anderes Spoolverzeichnis angegeben ist.

Im Verzeichnis /var/spool/mail sind die Postfächer aller Systembenutzer abgelegt. Jedes Postfach ist eine einfache Datei, in der alle persönlichen Briefe aneinandergehängt sind. Die Programme zur Mailverwaltung können mit diesen Dateien arbeiten, ohne sie durcheinander zu

bringen. Die Datensicherheit der persönlichen Post ist wie im Heimatverzeichnis durch die ausschließlich auf den Eigentümer begrenzten Zugriffsrechte weitgehend sichergestellt. Allein die Systemverwalterin (root) kann alle Dateien unabhängig von den Zugriffsrechten lesen.

---

[Next](#) [Up](#) [Previous](#) [Contents](#) [Index](#)

**Next:** [Von GNU's, Muscheln und](#) **Up:** [Reise durch's Dateisystem](#) **Previous:** [Das Verzeichnis /usr](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)



# Von GNU's, Muscheln und anderen Tieren

Das folgende Kapitel enthält Beschreibungen der wichtigsten Benutzerkommandos des Linux-Basissystems. Im darauffolgenden Kapitel werden die wichtigsten Kommandos für die Systemverwalterin erklärt. Beide Kapitel orientieren sich an den Manual-Pages, den Online-Hilfen, die mit den Programmen verteilt werden.

---

- [Intro\(1\) - oder die Erklärung der Erklärung](#)
  - [Syntax:](#)
  - [Beispiel:](#)
  - [Siehe auch:](#)
  - [Die 13 goldenen Regeln für ein gelungenes Kommando](#)
- [ar](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- [basename](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Siehe auch:](#)
- [bash](#)
  - [Funktion](#)
  - [Syntax](#)
  - [Beschreibung](#)
  - [Interaktive Shell und Shellprogrammierung](#)
  - [Der Kommandozeileneditor](#)
  - [Der Kommandozeilenspeicher \(history\)](#)
  - [Anpassung des Kommandozeileneditors](#)

- [Interpretation der Kommandozeile](#)
- [Kommentare](#)
- [Der Status](#)
- [Shell Grammatik](#)
- [Quotierung](#)
- [Ein-/Ausgabe-Umleitung](#)
- [Pipelines](#)
- [Hintergrundprozesse](#)
- [Listen](#)
- [Gruppen und Kontrollstrukturen: Blöcke, Schleifen, Verzweigungen, Funktionen](#)
- [Parameter](#)
- [Erweiterung](#)
- [Synonyme](#)
- [Signale](#)
- [Eingabeaufforderung](#)
- [Wenn alles getan ist](#)
- [Eingebaute Shellkommandos](#)
- [Login- und andere Shells](#)
- [Optionen](#)
- [Argumente beim Aufruf der Shell](#)
- [Dateien](#)

- [cat](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)
- [Beispiele:](#)

- [chgrp](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)
- [Siehe auch:](#)

- [chmod](#)

- [Funktion:](#)

- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)
- [Beispiel:](#)
- [Siehe auch:](#)

- [chsh](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)

- [cksum](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Siehe auch:](#)

- [cmp](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)
- [Siehe auch:](#)

- [comm](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)
- [Siehe auch:](#)

- [compress](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)

- [cp](#)

- [Funktion:](#)
- [Syntax:](#)
- [Optionen:](#)

- [cpio](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)
- [Siehe auch:](#)

- [csplit](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)
- [Beispiel:](#)
- [Siehe auch:](#)

- [cut](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)
- [Beispiel:](#)

- [date](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Die Systemzeit einstellen](#)
- [Optionen:](#)
- [Umgebung:](#)
- [Beispiel:](#)

- [dd](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)
- [Beispiel:](#)

- [df](#)

- [Funktion:](#)

- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)
- [dirname](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Siehe auch:](#)
- [doshell](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
- [du](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- [echo](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- [egrep](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Beispiel:](#)
  - [Siehe auch:](#)
- [elvis](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)

- [Überblick](#)
- [Der `visual mode`](#)
- [Der `colon mode`](#)
- [Reguläre Ausdrücke](#)
- [Die Optionen von elvis](#)
- [Zwischenspeicher \(Puffer\)](#)
- [elvrec](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
- [env](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- [expand](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- [expr](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
- [fdformat](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- [file](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- [find](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)
- [Tests:](#)
- [Aktionen:](#)
- [Operatoren:](#)
- [Beispiel:](#)
- [fold](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- [free](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- [grep](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Beispiel:](#)
- [groff](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Umgebung:](#)
  - [Dateien:](#)
  - [Beispiel:](#)
  - [Siehe auch:](#)
- [groups](#)
  - [Funktion:](#)

- [Syntax:](#)
- [Beschreibung:](#)
- [Siehe auch:](#)

- [gzip](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)
- [Siehe auch:](#)

- [head](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)

- [hexdump](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)
- [Siehe auch:](#)

- [hostname](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Siehe auch:](#)

- [id](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)

- [install](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)



- [join](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- [kill](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- [less](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Kommandos:](#)
  - [Optionen:](#)
  - [Umgebungsvariable](#)
  - [Siehe auch:](#)
- [ln](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- [login](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- [logname](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
- [ls](#)
  - [Funktion:](#)
  - [Syntax:](#)

- [Beschreibung:](#)
  - [Optionen:](#)
- [man](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- [mcopy](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- [mdel](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- [mdir](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- [mformat](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- [mkdir](#)
  - [Funktion:](#)
  - [Syntax:](#)

- [Optionen:](#)
- [Siehe auch:](#)

- [mkfifo](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)
- [Siehe auch:](#)

- [mmd](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)
- [Siehe auch:](#)

- [more](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)

- [mrd](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Siehe auch:](#)

- [mread](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)
- [Siehe auch:](#)

- [mt](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)

- [Beispiel:](#)
  - [Siehe auch:](#)
- [mtools](#)
  - [Funktion:](#)
  - [Beschreibung:](#)
- [mv](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- [newgrp](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
- [nice](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- [nl](#)
  - [Funktion](#)
  - [Syntax](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- [nohup](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
- [od](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)

- [Optionen:](#)
- [passwd](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Siehe auch:](#)
- [paste](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- [pr](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- [printenv](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
- [ps](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- [pwd](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
- [rm](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- [rmdir](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)

- [sed](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)

- [setfdprm](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)
- [Siehe auch:](#)

- [sleep](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)

- [sort](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)

- [split](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)

- [strace](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)

- [stty](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Beispiel:](#)
- [su](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- [sum](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- [superformat](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- [sync](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
- [tac](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- [tail](#)
  - [Funktion:](#)

- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)
- [Siehe auch:](#)

- [tar](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)
- [Beispiel:](#)
- [Siehe auch:](#)

- [tee](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)

- [touch](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)

- [tty](#)

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)

- [uname](#)

- [Funktion:](#)
- [Syntax:](#)
- [Optionen:](#)

- [uniq](#)

- [Funktion:](#)
- [Syntax:](#)
- [Optionen:](#)

- [wall](#)



- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [wc](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- [who](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- [write](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)

## Subsections

- [Syntax:](#)
  - [Beispiel:](#)
  - [Siehe auch:](#)
  - [Die 13 goldenen Regeln für ein gelungenes Kommando](#)
- 

# Intro(1) - oder die Erklärung der Erklärung

## Syntax:


Nach einer knappen, einzeiligen Darstellung der Funktion eines bestimmten Kommandos wird nach dem Untertitel „**Syntax:**“ der Aufruf auf der Kommandozeile mit allen Optionen dargestellt. Dazu wird eine strenge Form benutzt:

**name** [-Optionen ...] [*Argumente* ...]

### **name**

ist der Name des Kommandos. Das ist sowohl der vollständige Name der Datei, in der das Programm gespeichert ist als auch der Name, unter dem dieses Kommando aufgerufen werden muß. Besondere Endungen zur Kennzeichnung des Dateityps gibt es bei Linux nicht.

### **-Optionen**

sind „Befehle an das Programm“, mit denen dessen Verhalten manipuliert werden kann. Eine Option ist durch einen einzelnen Buchstaben bezeichnet.  Bei den Optionen kann zwischen „Schaltern“ und „Reglern“ unterschieden werden. Während Schalter allein durch ihr Auftreten bzw. durch ihre Abwesenheit auf der Kommandozeile bestimmte Ereignisse auslösen, muß zu den Reglern noch ein Argument angegeben werden. Allen Optionen gemeinsam ist, daß sie durch ein Minuszeichen eingeleitet werden.

Neben den Optionen, die nur durch einen Buchstaben bezeichnet werden, gibt es bei den GNU Utilities häufig ausdrücklich benannte Optionen. Diese Optionen werden durch zwei Minuszeichen eingeleitet. In der Kommandobeschreibung werden diese Optionen nicht mehr aufgeführt. Ihre Funktion wird in allen Fällen auch durch Buchstabenoptionen abgedeckt.

### *Argumente*

können entweder zu den oben erklärten regelnden Optionen gehören (Optionsargumente), oder sie können sich direkt auf das Kommando beziehen (Kommandoargumente, Operanden). In jedem Fall soll durch die kursive Schrift angedeutet werden, daß diese Argumente nicht wörtlich eingegeben werden sollen. Argumente dürfen nicht mit einem Minuszeichen beginnen.

Als Ausnahme ist als Kommandoargument meistens ein einzelnes Minuszeichen als Symbol für den Standardeingabekanal erlaubt.

{ Argument1,Argument2 }


ist die Darstellung für Optionsargumente mit einem eingeschränkten Wertebereich. Hier darf nur eines der zur Auswahl stehenden Argumente wörtlich eingesetzt werden.

[     ]

eckige Klammern schließen in der Regel Kommandoteile ein, die wahlfrei sind. Mit solchen Erweiterungen wird das Verhalten eines Kommandos bestimmt und verändert. In den wenigen Fällen, wo die eckigen Klammern selbst Teil des Kommandos sind (wie zum Beispiel beim `test`-Kommando), wird der andere Charakter dieser Klammer durch besonders fetten Druck dargestellt.

...

Wenn mehrere gleichartige Elemente eines Kommandos mehrfach vorkommen können, wird das durch Fortsetzungspunkte `...' gekennzeichnet.

Bestimmte Tasten werden in kleinen Großbuchstaben (KAPITÄLCHEN) benannt. Das sind vor allem das LEERZEICHEN oder SPACE, der TABULATOR oder TAB, der RÜCKSCHRITT oder BACKSPACE, das ZEILENENDE, das meist als RETURN, manchmal aber auch als NEWLINE  angesprochen wird, sowie die Umschalttasten ALT, CONTROL und ESCAPE.

Gelegentlich werden Tastenkombinationen CONTROL-Irgendetwas angesprochen. Dabei wird neben der ausgeschriebenen Tastenbezeichnung auch die Abkürzung CTRL oder die symbolische Darstellung durch ein Dach (Caret) `^' verwendet. Zum Beispiel steht ^D für CONTROL-D: das ist das EOF-Zeichen. Dieses Zeichen wird durch gemeinsames Drücken der CONTROL- und der `D'-Taste erzeugt.

## Beispiel:

Beispiele werden in Courier (Schreibmaschinenschrift) gegeben. Wenn ganze Kommandozeilen vorgeführt werden, wird der \$ Standardprompt der Shell dargestellt. Die Zeilen bis zum nächsten Prompt zeigen gegebenenfalls die Bildschirmausgabe des Beispiels:

```
$ echo 'Hallo Welt!'
': Event not found.
$ _
```

## Siehe auch:

Unter diesem Stichwort wird bei vielen Kommandobeschreibungen auf andere Stellen des Buches verwiesen, die weitere Information zum Thema bieten.

Zum Glück ist dieses Handbuch nicht die einzige Quelle gültiger Information. Ein umfangreiches Online-Hilfesystem bietet eine Vielzahl weiterer Antworten - wenn auch in der Regel in englischer Sprache. Am Ende vieler Kommandobeschreibungen in diesem Buch sind Verweise auf die Online-Hilfen gegeben.

Dabei kommen vor allem zwei Hilfe-Systeme in Frage:

- das `man`-Kommando, das zur Anzeige der klassischen Manual-Pages benutzt wird. Zu jedem Kommando gibt es solche „manpages“ (so sollte es zumindest sein). Die werden angezeigt,

indem das `man`-Kommando mit dem fraglichen Kommandonamen als Argument aufgerufen wird. Unter der grafischen Benutzeroberfläche von Linux, dem X Window System, gibt es zur Anzeige der Manual-Pages das mausgesteuerte `xman`-Kommando. Nähere Information zum `man`-Kommando gibt es [hier](#) des Handbuches; zu `xman` wird beim Programmstart automatisch ein Hilfstext angezeigt.

- das `info` Hilfesystem, dem die TEXinfo Dateien zugrunde liegen. Diese Hilfstexte sind so konzipiert, daß sie sowohl mit dem TEX Satzsystem formatiert und anschließend ausgedruckt werden, als auch, entsprechend aufbereitet, mit speziellen Programmen durchsucht und gelesen werden können. Als Programme kommen `info` bzw. sein grafisches Pendant `xinfo` oder der `emacs` Editor in Frage. Der Editor bietet einen speziellen Info-Modus, mit dem exakt die gleiche Funktionalität wie mit dem separaten `info` Programm erzielt wird.

Bei der Referenzierung der Online-Hilfen wird entweder im Falle der Manual-Pages die Sektion angegeben, also beispielsweise

**Siehe auch:**

`geqn(1)`, `gsoelim(1)`, `groTTY(1)`

für die Verwandten des `groff` aus Sektion 1, oder es steht anstelle der Sektion das Wort „`info`“ in den Klammern, was dann der entsprechende Verweis auf das TEXinfo-System ist.

## Die 13 goldenen Regeln für ein gelungenes Kommando

Für die Syntax eines Kommandos gibt es im POSIX 1003.2 Standard dreizehn Richtlinien. Diese Regeln sind nicht zwingend, es wird aber allen Autoren neuer Utilities empfohlen, sich daran zu halten. In einem Shellscript kann mit der Shellfunktion [getopts](#) eine nach den Regeln 3-10 eingegebene Kommandozeile analysiert werden. In C-Funktionen übernimmt `getopt(3)` diese Aufgabe.

1. Kommandonamen sollten zwei bis neun Zeichen lang sein.
2. Kommandonamen sollten nur aus ASCII-Kleinbuchstaben oder Ziffern bestehen.
3. Jede Optionsbezeichnung sollte aus einem einzelnen alphanumerischen Zeichen bestehen. Die Option `-W` soll für herstellerspezifische Erweiterungen reserviert sein.
4. Alle Optionen sollten von einem Minuszeichen `-` eingeleitet werden.
5. Optionen ohne Argumente (Schalter) sollten hinter einem einzigen Minuszeichen gruppiert werden können.
6. Jedes Optionsargument sollte von der Option, auf die es sich bezieht, durch (ein) Leerzeichen getrennt sein.
- 7.

Ein Optionsargument sollte nicht optional sein.

8.

Wenn mehrere Optionsargumente gleichzeitig erlaubt sind, sollten diese als ein einziges Kommandozeilenargument erscheinen. Dazu können die Optionsargumente entweder in Anführungszeichen eingeschlossen, oder, durch Komma getrennt, ohne Leerzeichen aufgelistet werden.

9.

Alle Optionen sollten vor den Kommandoargumenten angegeben werden.

10.

Zwei Minuszeichen - sollten als Markierung für das Ende der Kommandozeilenoptionen interpretiert werden. Alle folgenden Argumente sollten als Operanden für das Kommando behandelt werden, auch wenn sie mit einem Minuszeichen beginnen. Das - Symbol sollte nicht als Option oder Operand benutzt werden.

11.

Die Reihenfolge der Optionen untereinander sollte keine Rolle spielen, es sei denn, eine Option ist als ausschließlich und dominant dokumentiert. Solche Optionen können alle vorhergehenden inkompatiblen Optionen abschalten. Wenn ein Regler (Option mit Optionsargument) wiederholt wird, sollte die Interpretation in der Reihenfolge des Auftretens erfolgen.

12.

Die Reihenfolge der Kommandoargumente (Operanden) kann von Bedeutung sein, und positionsabhängige Interpretationen sind für ein Utility spezifisch.

13.

Für Utilities, die als Operanden Dateien benutzen, die sie zum Lesen oder Schreiben öffnen, sollte ein einzelnes, von Blanks eingeschlossenes Minuszeichen '-' den Standardeingabekanal bezeichnen. Wenn es aus dem Zusammenhang eindeutig hervorgeht, kann auch der Standardausgabekanal so bezeichnet werden.

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [ar](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [Von GNU's, Muscheln und](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

**Next:** [basename](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [Intro\(1\) - oder die](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# ar

## Funktion:

**ar** ist ein Werkzeug zum Erstellen und Verwalten von Archiven oder Bibliotheken, insbesondere für die Objektdateien des C-Compilers

## Syntax:

**ar** [dmpqrtx [abi *Positionsname* [cilosuv]]] *Archiv Datei* ...

## Beschreibung:

Mit **ar** werden beliebig viele Dateien zu einer einzigen Zusammengefaßt. Eine Kompression findet nicht statt. *Archiv* ist der Name der zu bearbeitenden Archivdatei. Die Archivdateien haben per Konvention die Endung ``.a'`. *Datei* ist der Name einer Datei, die in das Archiv eingefügt oder aus ihm gelöscht werden soll.

**ar** wird vor allem zur Verwaltung von Bibliotheken für den C-Compiler verwendet. Der Linker benötigt die spezielle Datei `___.SYMDEF`, in der die Symboltabellen aller Objektdateien zusammengefaßt sind. Mit der Option `-s` wird diese Datei erstellt und in das Archiv eingefügt.

## Optionen:

d

(delete) löscht *Datei* aus dem *Archiv*

m

(move) verschiebt die *Datei* nach (**a**fter) vor (**b**efore) oder ersetzt sie anstelle (**i**nstead) der Datei *Positionsname*

p

(print) gibt die *Datei* auf die Standardausgabe; die Zusatzoption **v** (verbose) gibt den Dateinamen vor jeder Datei aus

q

(quick) fügt die *Datei* in das *Archiv* ein, ohne zu prüfen, ob die Datei schon vorhanden ist; durch die Zusatzoption **c** wird die Warnung beim Erzeugen eines neuen Archivs unterdrückt

r

(replace) *Datei* ersetzt den gleichnamigen Eintrag in *archiv*; wenn noch kein Eintrag dieses Namens existiert wird, ein neuer Angelegt

t

(table) gibt das Inhaltsverzeichnis aus; die zusätzliche Option **v** läßt die ausführliche Version anzeigen

x

(extract) *Datei* wird aus dem *Archiv* herauskopiert

s

sorgt für Erstellung und Aktualisierung der Symboltabelle ``_ _'.SYMDEF` für den Linker, wie das Kommando `ranlib`

## Autor:

viele, Free Software Foundation

---

Next	Up	Previous	Contents	Index
------	----	----------	----------	-------

**Next:** [basename](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [Intro\(1\) - oder die](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

Next: [bash](#) Up: [Von GNU's, Muscheln und](#) Previous: [ar](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Siehe auch:](#)
- 

# basename

## Funktion:

**basename** liefert den Dateinamen ohne Pfadanteil

## Syntax:

**basename** *Name* [*Suffix*]

## Beschreibung:

**basename** schneidet bei einem Dateinamen mit absoluter Pfadangabe den Pfadanteil des Namens ab und liefert den bloßen Dateinamen. Bei optionaler Angabe einer Dateiendung (Suffix) wird auch diese abgeschnitten. **basename** wird vor allem bei der Shellprogrammierung verwendet.

## Siehe auch:

[dirname](#) und bei der [bash](#) und [bash](#)

## Autor:

Free Software Foundation



## Subsections

- [Funktion](#)
- [Syntax](#)
- [Beschreibung](#)
- [Interaktive Shell und Shellprogrammierung](#)
- [Der Kommandozeileneditor](#)
- [Der Kommandozeilenspeicher \(history\)](#)
  - [Der Kommandozeilenspeicher im Editor](#)
  - [History im C-Shell-Stil](#)
  - [Bezugnahme auf eine frühere Kommandozeile](#)
  - [Bezugnahme auf ein Wort einer früheren Kommandozeile](#)
  - [Modifikation der bezogenen Kommandozeilen](#)
  - [Beispiele für die History-Funktion](#)
- [Anpassung des Kommandozeileneditors](#)
  - [Bedingte Ausführung von `. inputrc`](#)
  - [Beispiel:](#)
  - [Der erweiterte Editor: sekundärer Prompt](#)
- [Interpretation der Kommandozeile](#)
- [Kommentare](#)
- [Der Status](#)
- [Shell Grammatik](#)
  - [Reservierte Wörter](#)
  - [Atome, Wörter, Token](#)
  - [Kommandos - nicht unbedingt einfach](#)
- [Quotierung](#)
- [Ein-/Ausgabe-Umleitung](#)
  - [Eingabeumleitung](#)
  - [Ausgabeumleitung](#)
  - [Anfügen der Ausgabe an eine existierende Datei](#)
  - [Zusammenfassung der Standardausgabe mit der Standardfehlerausgabe](#)
  - [Shellscript-Dokumente](#)
  - [Verdoppelung der Dateikennung](#)
  - [Öffnen einer Datei zum Lesen und Schreiben](#)

- [Pipelines](#)
- [Hintergrundprozesse](#)
- [Listen](#)
  - [Listen mit ; und &](#)
  - [Bedingte Ausführung](#)
- [Gruppen und Kontrollstrukturen: Blöcke, Schleifen, Verzweigungen, Funktionen](#)
- [Parameter](#)
  - [Shell- und Umgebungsvariable](#)
  - [Eindimensionale Arrays](#)
  - [Positionsparameter](#)
  - [Spezialparameter](#)
- [Erweiterung](#)
  - [Klammererweiterung](#)
  - [Tildenerweiterung](#)
  - [Parametererweiterung](#)
  - [Kommandosubstitution](#)
  - [Arithmetische Erweiterung](#)
  - [Prozeßsubstitution](#)
  - [Worttrennung](#)
  - [Pfadnamenerweiterung](#)
  - [Quotenreduktion](#)
- [Synonyme](#)
- [Signale](#)
- [Eingabeaufforderung](#)
- [Wenn alles getan ist](#)
- [Eingebaute Shellkommandos](#)
  - [:](#)
  - [alias](#)
  - [bg](#)
  - [bind](#)
  - [break](#)
  - [builtin](#)
  - [bye](#)
  - [cd](#)
  - [command](#)

- [continue](#)
- [declare](#)
- [dirs](#)
- [disown](#)
- [echo](#)
- [enable](#)
- [eval](#)
- [exec](#)
- [exit](#)
- [export](#)
- [fc](#)
- [fg](#)
- [getopts](#)
- [hash](#)
- [help](#)
- [history](#)
- [jobs](#)
- [kill](#)
- [let](#)
- [local](#)
- [logout](#)
- [popd](#)
- [pushd](#)
- [pwd](#)
- [read](#)
- [readonly](#)
- [return](#)
- [set](#)
- [shift](#)
- [shopt](#)
- [source](#)
- [suspend](#)
- [test](#)
- [time](#)
- [times](#)

- [trap](#)
  - [type](#)
  - [typeset](#)
  - [ulimit](#)
  - [umask](#)
  - [unalias](#)
  - [unset](#)
  - [wait](#)
  - [Login- und andere Shells](#)
  - [Optionen](#)
  - [Argumente beim Aufruf der Shell](#)
  - [Dateien](#)
- 

# bash

## Funktion


**bash** - die Wiedergeburtsmuschel. Stark entwickelter Nachfahre eines unsterblichen Wesens aus dem Unix (Kreidezeit).

## Syntax

**bash** [*Optionen*] [*Datei*]

## Beschreibung

In der Computersteinzeit wurden Programme in Lochkarten gestanzt und zusammen mit den Daten in mechanische Kartenleser gepackt. Die Auswahl zwischen den möglichen Programmen fand also nicht an einer Systemconsole, sondern vor den Regalen des Kartenarchivs statt. Heute sind die Programme und die Daten auf magnetischen Datenträgern gespeichert. Die Festplatte einer durchschnittlichen Linux-Installation bietet mindestens 300 verschiedene Programme zur Auswahl.

Die Frage, wie die Auswahl eines konkreten Programms stattfindet, ist ebenso trivial wie bodenlos tiefgründig. Die Antwort ist ein Metaprogramm, das automatisch geladen wird und dessen Hauptaufgabe es ist, weitere Programme zu laden. Unter UNIX und seinen Verwandten wird so ein Programm als Shell bezeichnet. Sie hat, im Gegensatz zu vielen vergleichbaren Programmen anderer Betriebssysteme, den Status eines Benutzerprogramms und kann deshalb nach Belieben ausgetauscht werden. 

An der Benutzeroberfläche entfesselt sich leicht eine Art Glaubenskrieg zwischen den Protagonisten unterschiedlicher Modelle. Zwischen den grafischen (mausgesteuerten, bildschirmorientierten) und

den textuellen (tastaturgesteuerten, zeilenorientierten) Benutzeroberflächen scheinen sich die Geister zu scheiden. Die Vorteile der grafischen Benutzerführung liegt vor allem in ihrer leichten Erlernbarkeit. Durch die Präsentation der möglichen Aktionen in Menüs kann der ungeübte Benutzer intuitiv den Weg zu seiner Problemlösung finden. Zeilenorientierte Oberflächen, sogenannte Kommandozeileninterpreten, haben ihren Vorteil in der Vielseitigkeit. Während in einem Menüsystem zwangsläufig nur die Funktionen zu erreichen sind, für die es Menüeinträge gibt, können im Kommandozeileninterpreter alle Kommandos mit allen zulässigen Optionen aufgerufen werden. Genau diese Vielseitigkeit macht den Kommandozeileninterpreter - die Shell - zu einem unverzichtbaren Werkzeug der Systemverwalterin, das auch in kommerziellen Systemen mit menügesteuerter Administratorshell nicht ersetzt werden kann.

Die „Standardshell“ von AT&T Unix ist die nach ihrem Entwickler Steven R. Bourne benannte Shell (die unter der Kommandozeichnung `sh` aufgerufen wird). Neben ihren Diensten als interaktiver Kommandozeileninterpreter bietet die Bourne-Shell noch eine mächtige Sprache zum Erstellen von Shellprogrammen. Solche Shellscripts erlauben schnell und unkompliziert die Zusammenfassung immer wiederkehrender Kommandofolgen als Batchdatei. Die Möglichkeiten der Shellprogrammierung gehen aber noch viel weiter, wie die weiter hinten folgende Beschreibung zeigen wird.

Neben der „alten“ Bourne-Shell gibt es noch eine ganze Reihe weiterer Shells. Andere bekannte Shells sind die an der Berkeley Universität entwickelte `csh` und die nach ihrem Entwickler David Korn benannte Shell `ksh`. Die Shells unterscheiden sich vor allem in den Shellsprachen, sie haben aber auch neue Eigenschaften zur Erleichterung der interaktiven Benutzung als Kommandozeileninterpreter. Die „Bourne Again Shell“ `bash` vereint die altbekannte und bewährte Shellsprache der Bourne Shell mit den fortschrittlichen und beliebten Interaktionsfunktionen der anderen Shells. Die `bash` ist als Standardshell in allen Linux-Distributionen enthalten.

Es ist erklärter Anspruch der `bash`, zur Standard-Bourne-Shell kompatibel zu sein, und Ziel ist die volle Übereinstimmung mit dem POSIX-1003.2-Standard. Die Kompatibilität zur Bourne-Shell ist besonders wichtig, um die vielen Shellscripts für diese Standardshell auch mit der `bash` benutzen zu können.

## Interaktive Shell und Shellprogrammierung

Die alltägliche Arbeit mit der Shell findet interaktiv statt. Das heißt, die Shell gibt eine Eingabeaufforderung (Prompt) aus, die vom Benutzer mit einer Folge von Tastatureingaben, der Kommandozeile, beantwortet wird. Eine Kommandozeile wird durch ein Zeilenendezeichen (RETURN) abgeschlossen. Die Shell interpretiert daraufhin die eingegebene Zeile und führt die gegebenenfalls darin formulierten Kommandos aus. Normalerweise gibt die Shell nach der vollständigen Bearbeitung der Kommandozeile wieder eine Eingabeaufforderung aus, um den gleichen Vorgang erneut einzuleiten.


Für die Shell macht es wenig Unterschied, ob sie eine Kommandozeile direkt von der Tastatur liest, oder ob sie die gleiche Zeile aus einem „eingefrorenen Datenstrom“, einer Datei, erhält. Für den Anwender ist es enorm praktisch, immer wiederkehrende Kommandofolgen in einer Textdatei zusammenzufassen und dann diese Datei anstelle der Tastatureingabe bearbeiten zu lassen. Genau das ist der Ursprung der Shellprogrammierung.


In der Realität bieten alle Unix-Shells Programmiersprachen, die weit über diese Stapelverarbeitung (sogenannte Batch-Jobs) hinausgehen. Es können Variable benutzt werden, Verzweigungen und Schleifen sind möglich, es können sogar Script-Funktionen definiert werden, die eine strukturierte Programmierung wie in den bekannten Hochsprachen erlauben.


Nun besteht umgekehrt kein vernünftiger Grund, weshalb die Elemente der Shellprogrammierung nicht auch auf der Kommandozeile verwendet werden sollten. Natürlich wird niemand ein zig-zeiliges Shellprogramm direkt auf der Kommandozeile eingeben. Andererseits macht es wenig Sinn, für jede vierzeilige `for`-Schleife ein Shellsript zu schreiben.

Damit wird klar, daß die Grenzen zwischen interaktiver Benutzung und Shellprogrammierung fließend sind. Je nach Blickwinkel werden aber die Schwerpunkte anders gesetzt. Diese Beschreibung richtet sich in erster Linie an interaktive Benutzer. Sie erhebt aber auch den Anspruch auf Vollständigkeit. Um den Text einigermaßen lesbar zu machen, sind die Abschnitte mit ausgesprochen shellscriptspezifischen Inhalten in einem kleineren Schriftgrad gedruckt. Solche Textteile können beim ersten Lesen übersprungen werden.

## Der Kommandozeileneditor

Alle Tastatureingaben, die zeilenweise erfolgen, können mit bestimmten Steuerzeichen „editiert“ werden. Mit dem Steuerzeichen `CONTROL-U` wird normalerweise die komplette Zeile gelöscht, `CONTROL-C` bricht die Eingabe ab, `CONTROL-D`  steht für das Dateiende (EOF, End Of File), also für das Ende der interaktiven Eingabe insgesamt, `CONTROL-H` oder `BACKSPACE` löscht das letzte Zeichen.

Diese primitiven Funktionen werden vom Terminaltreiber des Kernels im „cooked“-Modus direkt angeboten. Die `bash` bietet darüber hinaus einen Kommandozeileneditor, der in Umfang und Funktion den Standardeditoren `vi` und `emacs` nachempfunden ist.  Sie können die Funktionen des Kommandozeileneditors Ihren eigenen Bedürfnissen [anpassen](#). Zwischen den Standardbelegungen kann mit dem [set-Shellkommando](#) umgeschaltet werden. Normalerweise arbeitet der Kommandozeileneditor im `emacs`-Modus.

Die Editorbefehle im `emacs`-Modus benutzen zwei Sondertasten: die `CONTROL-` und die Metataste. Die `CONTROL`-Taste ist auf der PC-Tastatur in der linken unteren Ecke (manchmal `STRG`). Eine Metataste ist unter diesem Namen in der Regel nicht vorhanden. Meistens wird die linke `ALT`-Taste mit dieser Funktion belegt. Wenn das nicht der Fall ist, kann die `ESCAPE`-Taste (`ESC`) benutzt werden. Während `CONTROL` und `ALT` gemeinsam mit dem Buchstaben gedrückt werden müssen, wird `ESC` vor dem Buchstaben gedrückt (zwei Anschläge). 

In den Erklärungen wird `CONTROL` mit ``C-'` und die Metataste mit ``M-'` gekennzeichnet. In manchen Fällen müssen `CONTROL` und `ALT` zusammen gedrückt werden, das wird dann durch `M-C-` symbolisiert.

## Positionieren der Einfügemarke

### Zeilenanfang (C-a)

setzt die Einfügemarke an den Anfang der Kommandozeile.  
(beginning-of-line)

### Zeilenende (C-e)

setzt die Einfügemarke an das Ende der Kommandozeile. (end-of-line)

## **Zeichen vorwärts (C-f)**

setzt die Einfügemarke ein Zeichen nach rechts. Diese Funktion ist auch mit der rechten Pfeiltaste im Cursorblock erreichbar. (forward-char)

## **Zeichen rückwärts (C-b)**

setzt die Einfügemarke ein Zeichen nach links. Diese Funktion ist auch mit der linken Pfeiltaste im Cursorblock erreichbar. (backward-char)

## **Wort vorwärts (M-f)**

setzt die Einfügemarke an das Ende des aktuellen Wortes oder ein Wort weiter (wenn die Einfügemarke zwischen zwei Wörtern steht). (forward-word)

## **Wort rückwärts (M-b)**

setzt die Einfügemarke an den Anfang des aktuellen Wortes oder des vorhergehenden Wortes (wenn die Einfügemarke zwischen zwei Wörtern steht). (backward-word)

## **Bildschirm löschen (C-l)**


löscht alle Zeichen vom Bildschirm. Die Einfügemarke erscheint danach auf der ersten Zeile. (clear-screen)

# **Automatische Erweiterung von Kommando- und Dateinamen**

## **erweitern (TAB)**

versucht die bis zu diesem Editorbefehl geschriebene Kommandozeile sinnvoll zu ergänzen. Das geschieht, indem die Zeichenfolge des letzten Wortes bis zur Einfügemarke mit den Namen von Kommandos, Dateien und Verzeichnissen verglichen und das Wort so weit ergänzt wird, wie eine eindeutige Zuordnung möglich ist. Das erste Wort eines einfachen Kommandos wird dabei nur mit den Kommandonamen in den PATH-Verzeichnissen verglichen, die weiteren Wörter eines einfachen Kommandos werden dagegen nur noch mit Datei- und Verzeichnisnamen im aktuellen Verzeichnis (Arbeitsverzeichnis) verglichen. In einigen Fällen (z. B. bei Pipelines) werden auch die ersten Wörter der folgenden Kommandos auf diese Weise richtig zugeordnet.

Probieren Sie diese Funktion des Kommandozeileneditors einfach aus. Sie werden schnell merken, wie mächtig und vielseitig sie ist.

Wenn ein Wort mehreren bekannten Namen zugeordnet werden kann, wird es nur so weit ergänzt, wie sich die Namen nicht unterscheiden. Wenn dann ein zweites Mal TAB gedrückt wird, werden alle erkannten Möglichkeiten angezeigt.  (complete)

## **mögliche Erweiterungen (M-?)**

zeigt alle als sinnvoll erkannten Erweiterungen an. Als sinnvoll gilt hier wieder ein Kommandoname als erstes Wort eines einfachen Kommandos und ein Datei- oder Verzeichnisname für jedes weitere Wort. (possible-completions)

## **nur Kommandoerweiterung (M-!)**

versucht, das Wort vor dem Cursor zu einem Kommandonamen zu erweitern. (complete-command)

## **mögliche Kommandoerweiterungen (C-x !)**

gibt eine Liste aller als mögliche Erweiterungen für das Wort vor der Einfügemarke in Frage kommenden Kommandonamen aus. (possible-command-completions)

### **nur Dateinamenerweiterung (M-/)**

vergleicht das Wort bis zum Cursor nur mit Datei- und Verzeichnisnamen. Werden passende Namen gefunden, findet eine Erweiterung wie beim TAB- Editorkommando statt.  
(complete-filename)

### **mögliche Dateinamen (C-x /)**

zeigt alle als sinnvoll erkannten Datei- und Verzeichnisnamen, ohne eine Erweiterung auszuführen. (possible-filename-completions)

### **Benutzernamenerweiterung (M-~)**

versucht, das Wort bis zur Einfügemarke zu einem Benutzernamen zu erweitern.  
(complete-username)

### **mögliche Benutzernamen (C-x ~)**

zeigt alle möglichen Benutzernamen, auf die das Wort bis zur Einfügemarke paßt.  
(possible-username-completions)

### **Variablenerweiterung (M-\$)**

versucht, das Wort bis zur Einfügemarke zu einem Variablennamen zu erweitern.  
(complete-variable)

### **mögliche Variable (C-x \$)**

zeigt alle möglichen Variablennamen an, auf die das Wort bis zur Einfügemarke paßt.  
(possible-variable-completion)

### **nur Hostnamenerweiterung (M-@)**

erweitert das Wort unter der Einfügemarke zu einem Hostnamen. Dieser Name wird aus der Datei /etc/hosts genommen, wenn in der Shellvariablen hostname\_completion\_file keine andere Datei angegeben ist. (complete-hostname)

### **mögliche Hostnamen (C-x @)**

zeigt alle möglichen Erweiterungen für den Hostnamen.  
(possible-hostname-completions)

### **Erweiterung aus der History (M-TAB)**


versucht, das Wort vor dem Cursor aus dem Historyspeicher zu ergänzen.  
(dynamic-complete-history)

### **Erweiterung in Klammern (M-{)**

schließt die Liste der möglichen Dateinamenerweiterungen in Klammern ein, so daß die Shell sie zu einer Liste dieser Namen erweitern kann. (complete-into-braces)

## **Änderungen des Textes**

### **Zeichen löschen (C-d)**

löscht das Zeichen unter dem Cursor. Wenn CONTROL-d als erstes Zeichen einer leeren Kommandozeile eingegeben wird, erzeugt diese Tastenkombination ein EOF-Zeichen für das Ende der interaktiven Eingabe.  (delete-char)

### **letztes Zeichen löschen (BACKSPACE)**

löscht das Zeichen vor dem Cursor. (backward-delete-char)

### **wörtlich einfügen (C-q, C-v)**

wird benutzt um CONTROL-Zeichen in die Kommandozeile zu schreiben, die normalerweise durch



den Editor abgefangen und bearbeitet werden. (quoted-insert)

### **Tabulator einfügen (M-TAB)**

schreibt ein TAB in die Kommandozeile. (tab-insert)

### **Text einfügen (a, b, A, 1, ?, ...)**

schreibt die Tastatursymbole (Buchstaben) in die Kommandozeile, wie sie von der Tastatur gelesen werden. Hinter dieser etwas verklausulierten Formulierung verbirgt sich die Eingabe eines normalen Buchstabens. (self-insert)

### **zwei Zeichen vertauschen (C-t)**

vertauscht das Zeichen vor der Einfügemarke mit dem Zeichen unter der Einfügemarke. Die Einfügemarke wandert außerdem um eine Stelle weiter. Wenn die Einfügemarke am Zeilenende angekommen ist, werden die beiden Zeichen vor der Einfügemarke vertauscht.

(transpose-chars)

### **zwei Wörter vertauschen (M-t)**

vertauscht das Wort unter der Einfügemarke mit dem Wort vor der Einfügemarke bzw. das Wort nach der Einfügemarke mit dem Wort vor der Einfügemarke. Die Einfügemarke steht danach hinter dem letzten der vertauschten Wörter. Wenn die Einfügemarke bereits am Ende der Kommandozeile steht, werden die beiden Wörter vor der Einfügemarke vertauscht.

(transpose-words)

### **GROSSBUCHSTABEN (M-u)**

wandelt alle Buchstaben des Wortes von der Einfügemarke an in Großbuchstaben um.

(upcase-word)

### **kleinbuchstaben (M-l)**

wandelt alle Buchstaben des Wortes von der Einfügemarke an in Kleinbuchstaben um.

(downcase-word)

### **Großschreibung (M-c)**

wandelt den Buchstaben unter der Einfügemarke oder den ersten Buchstaben des folgenden Wortes in Großbuchstaben um, die folgenden Buchstaben bis zum Wortende alle in Kleinbuchstaben. Anschließend steht die Einfügemarke hinter dem umgewandelten Wort.

(capitalize-word)

## **Ausschneiden und Einfügen**

### **bis zum Zeilenende ausschneiden (C-k)**

löscht die restliche Kommandozeile von der Einfügemarke an und speichert sie im Ausschneide-Ringspeicher. (kill-line)

### **vom Zeilenanfang ausschneiden (normalerweise ohne Belegung)**

schneidet den Anfang der Kommandozeile bis zur Einfügemarke aus. Dieses Kommando ist normalerweise nicht mit einer Tastenkombination verbunden. (backward-kill-line)

### **Rest des Wortes ausschneiden (M-d)**

schneidet von der Einfügemarke an ein Wort aus und speichert es im Ringspeicher. Als Wort werden hier alle Zeichen von der Einfügemarke an bis zum nächsten Sonderzeichen betrachtet. Eine beliebig lange Folge von Sonderzeichen unter oder unmittelbar nach der Einfügemarke wird nicht als Worttrenner behandelt. (kill-word)

### **Anfang des Wortes ausschneiden (M-BACKSPACE)**

schneidet alle Zeichen vor der Einfügemarke bis zum nächsten Sonderzeichen aus und speichert sie im Ringspeicher. Das Zeichen unter der Einfügemarke wird nicht ausgeschnitten. Eine beliebig lange Folge von Sonderzeichen unmittelbar vor der Einfügemarke wird nicht als Worttrenner behandelt. (`backward-kill-word`)

### **vom Zeilenanfang bis zur Einfügemarke ausschneiden (C-u)**

schneidet den Anfang der Kommandozeile bis zur Einfügemarke aus und speichert ihn im Ringspeicher. Das Zeichen unter der Einfügemarke wird nicht ausgeschnitten. (`unix-line-discard`)

### **Anfang des Wortes ausschneiden (C-w)**

schneidet alle Zeichen vor der Einfügemarke bis zum nächsten Leerzeichen aus und speichert sie im Ringspeicher. (`unix-word-rubout`)

### **ausgeschnittenes Stück einfügen (C-y)**

fügt das zuletzt in den Ausschneide-Ringspeicher geschriebene Stück vor der Einfügemarke ein. Das Stück wird dabei aus dem Ringspeicher entfernt. (`yank`)

### **Ausschneide-Ringspeicher rotieren (M-y)**

ersetzt das zuletzt eingefügte Stück durch das vor diesem Stück in den Ringspeicher eingefügte Stück der Kommandozeile. Das zuvor ersetzte Stück wird wieder in den Ringspeicher eingefügt, das aktuell ersetzte Stück daraus entfernt. (`yank-pop`)

## **Argumente**

### **numerisches Argument (M-0, M-1, ... , M--)**

wird als numerisches Argument für das folgende Editorkommando benutzt. In der Regel wird durch dieses Argument die Anzahl der Wiederholungen dieses Editorkommandos bestimmt. (`digit-argument`)

### **universelles Argument (nicht Belegt)**

entspricht dem gleichnamigen Tastaturbefehl vom emacs-Editor. Es ist allerdings keiner Tastenkombination zugeordnet. (`universal-argument`)

## **Verschiedenes**

### **einlesen der Initdatei (C-x C-r)**

veranlaßt die Shell, die `.inputrc`-Datei neu einzulesen. (`re-read-init-file`)

### **Metazeichen (ESC)**

veranlaßt die Shell, das nächste Zeichen als Metazeichen zu interpretieren. Wichtig, wenn keine Metataste auf der Tastatur ausgewiesen ist: bei einer PC-Tastatur ist meist die linke ALT-Taste mit der Metafunktion belegt! (`prefix-meta`)

### **Kommando zurück (C-\_)**

nimmt das zuletzt ausgeführte Editorkommando zurück. Es werden alle für eine Kommandozeile gegebenen Editorkommandos gespeichert und sind auf diese Weise nacheinander reversibel. (`undo`)

### **alles zurück (M-r)**

stellt den ursprünglichen Zustand der Kommandozeile wieder her, nimmt also alle Editorkommandos zurück. (`revert-line`)

### **emacs Modus (C-e)**

schaltet aus dem vi-Modus in den emacs-Modus. (emacs-editing-mode)

# Der Kommandozeilenspeicher (history)


Die bash speichert eine gewisse Anzahl kompletter Kommandozeilen ab und erlaubt dem Benutzer, den Kommandozeilenspeicher auf zwei verschiedene Weisen anzusprechen.

Die Anzahl der gespeicherten Kommandozeilen kann mit der Shellvariablen HISTSIZE eingestellt werden. Weitere Einstellungen können mit den Variablen HISTFILE HISTFILESIZE und history\_control vorgenommen werden. Die Benutzung dieser Variablen wird [hier](#) beschrieben.

## Der Kommandozeilenspeicher im Editor

Im Gegensatz zu den Vorbildern emacs oder vi ist für den Kommandozeileneditor der Arbeitsbereich auf die aktuelle Zeile beschränkt. Absätze und Seiten existieren hier nicht. Mit dem Kommandozeilenspeicher eröffnet sich aber doch eine zweite Dimension, die das Blättern und Suchen in alten Zeilen zu einer praktischen Erweiterung des Editors macht.

### Abschluß einer Kommandozeile (RETURN)

Wie bereits gesagt, wird jede Kommandozeile mit einem Zeilenende (RETURN) abgeschlossen. Dabei ist es egal, wo sich die Einfügemarke innerhalb der Zeile gerade befindet, es wird immer die gesamte Zeile bearbeitet. Wenn sie nicht leer ist, wird sie sofort im „rohen“ Zustand in den Kommandozeilenspeicher übernommen.  Wenn die Shellvariable history\_control das Wort 'ignorespace' enthält, werden Kommandozeilen, die mit einem Leerzeichen beginnen, nicht gespeichert. Enthält die Variable das Wort 'ignoredups', werden nur die Kommandozeilen gespeichert, die sich von der zuletzt gespeicherten unterscheiden. Mit 'ignoreboth' werden die beiden Features eingeschaltet.

Wie alle Funktionen des Editors kann diese Funktion auf beliebige Tasten gelegt werden, wenn auch in diesem Fall davon abzuraten ist. (accept-line)

### Kommando zurück (C-p)

blättert im Kommandozeilenspeicher eine Position zurück und gibt die komplette Zeile zum Editieren auf den Bildschirm. Diese Funktion ist auch an die HOCH-Taste des Cursorblocks gebunden. (previous-history)

### Kommando vorwärts (C-n)

blättert im Kommandozeilenspeicher eine Position vorwärts und gibt die komplette Zeile zum Editieren auf den Bildschirm. Diese Funktion ist auch über die RUNTER-Taste des Cursorblocks zu erreichen. (next-history)

### zum ersten Kommando (M-<)

holt die erste Kommandozeile aus dem Kommandozeilenspeicher in den Editor. (beginning-of-history)

### zum letzten Kommando (M->)

holt die letzte Kommandozeile aus dem Kommandozeilenspeicher in den Editor. (end-of-history)

### Kommando inkrementell rückwärts suchen (C-r)

sucht schrittweise (inkrementell)rückwärts im Kommandozeilenspeicher nach einer Kommandozeile mit exakt passendem Muster. Das gesuchte Muster wird interaktiv eingegeben und die erste passende Zeile sofort angezeigt. Reguläre Ausdrücke im Muster werden nicht interpretiert, sondern buchstäblich mit den gespeicherten Kommandozeilen verglichen. Das Suchmuster kann mit BACKSPACE schrittweise zurückgenommen werden; das erste andere Editorkommando während der interaktiven Eingabe des Suchmusters beendet die Suche. (reverse-search-history)

### **Kommando rückwärts suchen (C-p)**

sucht nach der vollständigen Eingabe einer Zeichenkette die erste darauf passende Kommandozeile in der History. (non-incremental-reverse-search-history)

### **Kommando inkrementell vorwärts suchen (C-s)**

sucht mit jedem weiteren eingegebenen Buchstaben schrittweise (inkrementell) vorwärts im Kommandozeilenspeicher von der aktuellen Zeile an nach einer Kommandozeile mit einem bestimmtem Muster. Die Bearbeitung des Musters erfolgt in der gleichen Weise wie beim reverse-search-history-Editorkommando. (forward-search-history)

### **Kommando vorwärts suchen (M-n)**

sucht nach der Eingabe einer Zeichenkette vorwärts in der History nach der ersten passenden Kommandozeile. (non-incremental-forward-search-history)

### **Kommandozeile erweitern (M-C-e)**

führt die weiter unten beschriebene History-Expansion im C-Shell-Stil und die Synonymerweiterung aus, ohne die Kommandozeile auszuführen. Die Zeile kann daraufhin weiter bearbeitet werden. (expand-line)

### **letztes Argument einfügen (M-., M-\_)**

fügt das letzte Argument des letzten Kommandos an der Einfügemarke ein. (insert-last-argument)


### **n-tes Argument einfügen (M-C-y)**

fügt das n-te Argument der letzten Kommandozeile beim Cursor ein. Die Zahl wird als Argument (siehe oben) dem Kommando vorangestellt, Defaultwert ist 1. (yank-nth-arg)

### **ausführen und nächstes Kommando (C-o)**

führt das mit den Cursortasten in die Kommandozeile zurückgeholte Kommando aus und gibt nach dessen Beendigung automatisch die auf die ausgeführte Zeile folgende Zeile aus der History aus. (operate-and-get-next)

## **History im C-Shell-Stil**

Zusätzlich zu den oben beschriebenen Tastenfunktionen des Editors kann der Kommandozeilenspeicher mit einem History-Mechanismus benutzt werden, dessen Verwendung der C-Shell nachempfunden ist.  Diese History-Expansion kann mit dem Shellkommando `set +H` ausgeschaltet werden.

In der C-Shell ist es möglich, einzelne Wörter aus bestimmten Kommandozeilen im Kommandozeilenspeicher in die aktuelle Kommandozeile zu integrieren. Dazu wird zuerst eine Kommandozeile referenziert und danach ein Wort in dieser Kommandozeile angesprochen.

# Bezugnahme auf eine frühere Kommandozeile

Zum Auswählen einer Kommandozeile im Kommandozeilenspeicher gibt es verschiedene Möglichkeiten. Prinzipiell leitet ein Ausrufezeichen `!` die History-Substitution ein. Nur wenn das Ausrufezeichen nach einem Fluchtsymbol `\` steht oder von einem SPACE, RETURN, TAB, `=` oder `( ` gefolgt wird, findet die Substitution nicht statt.

Anstelle des Ausrufezeichens kann in der Shellvariablen `histchars` auch ein anderes Zeichen mit dieser Funktion belegt werden.

**!!**

wählt die letzte Zeile im Kommandozeilenspeicher.

**!*n***

wählt die Zeile Nummer *n*.

**!-*n***

wählt die aktuelle Zeile minus *n*

**!*Zeichenkette***

wählt die letzte Kommandozeile, deren Anfang mit der *Zeichenkette* übereinstimmt.

**!?*Zeichenkette*[?]**

wählt die letzte Kommandozeile, in der die *Zeichenkette* an irgendeiner Position vorkommt.

**^Alt^*Neu***

ist eine Abkürzung für `!!:s^Alt^Neu^` zur Ersetzung der Zeichenkette *Alt* durch *Neu* in der letzten Kommandozeile. Diese Konstruktion muß allein auf der Kommandozeile stehen.

**!#**

steht für die aktuelle Kommandozeile bis hier hin.

# Bezugnahme auf ein Wort einer früheren Kommandozeile

Auf die Kommandozeilenreferenz kann eine Wortreferenz folgen. Diese Wortreferenz wird durch einen Doppelpunkt `:` von der Zeilenreferenz getrennt. Wenn die Wortreferenz mit einem `^`, `\$`, `\*` oder `%` beginnt, kann der Doppelpunkt auch weggelassen werden. Die Wörter einer Kommandozeile sind vom Zeilenanfang an mit Null beginnend nummeriert.

***n***

bezeichnet das *n*te Wort. Das Wort 0 ist in der Regel der Kommandoname.

**^**

(Caret) steht für das erste Argument (das ist Wort Nummer 1).

**\$**

bezeichnet das letzte Argument.

**%**

steht für das bei *?Zeichenkette?* gefundene Wort.

***n-m***

ist der Bereich vom *n*ten bis zum *m*ten Wort

**\***

steht für alle Argumente, also die Wörter 1-\$. Wenn das Kommando nur aus einem Wort besteht, wird die leere Zeichenkette zurückgeliefert.

## Modifikation der bezogenen Kommandozeilen

Nach (oder anstelle) der Wortreferenz können noch ein oder mehrere Zeichen zur Modifikation des bezogenen Wortes folgen. Diese Zeichen werden wieder durch Doppelpunkte getrennt.

h

schneidet alle Zeichen nach dem letzten Slash von dem referenzierten Bereich der Kommandozeile ab. Die Funktion ist in gewisser Weise mit dem Shellutility `dirname` vergleichbar.

r

schneidet eine Endung der Form ``.xxx'` nach dem letzten Punkt ab.

e

läßt nur die Endung in der oben beschriebenen Form übrig.

t

entfernt aus dem referenzierten Bereich der Kommandozeile alles bis zum letzten Slash. Die Funktion ist dem Shellutility `basename` vergleichbar.

p

zeigt die entstandene Kommandozeile sofort an, ohne sie auszuführen.

[g]s/**Alt/Neu** [/]

ersetzt die Zeichenkette *Alt* bei ihrem ersten Auftreten durch *Neu*. Durch den Zusatz `g` kann erreicht werden, daß alle Vorkommen von *Alt* ersetzt werden. Anstelle der Slashes können beliebige andere Zeichen benutzt werden.

## Beispiele für die History-Funktion

Auf den ersten Blick ist die C-Shell History-Funktion wahrscheinlich etwas spröde. Die folgenden Beispiele zeigen einige Vorteile dieser Funktion bei der täglichen Arbeit mit der C-Shell.

Häufig wird eine Datei in ein Verzeichnis verschoben und unmittelbar anschließend das aktuelle Verzeichnis dorthin gewechselt.

```
$ mv broo.fazz.tar.gz /ftp/pub/comp/i386/Linux/utls/misc
$ cd !$
$ _
```

Mit ``!$'` wird das letzte Wort der vorhergehenden Kommandozeile eingefügt. Das gleiche Ergebnis wird auch durch die `META-.-` Funktion des Kommandozeileneditors ausgeführt.

Gelegentlich braucht man nicht das komplette Argument, sondern nur einen Teil davon:

```
$ less /usr/X386/include/X11/Xaw/SmeBSBP.h
$ find /usr/src/X11 -name !:t
find /usr/src/X11 -name SmeBSBP.h
find: /usr/src/X11: No such file or directory
$ _
```

Der Operator `! : t` liefert `SmeBSBP`. h. Hier kann die Referenz auf das letzte Argument weggelassen werden, weil die `: t`-Operation nur die Zeichen nach dem letzten Slash liefert.

Sehr nützlich ist auch die schnelle Substitution einer Zeichenfolge:

```
$ tar -tvzf /ftp/pub/Incoming/das.ist.neu.tar.gz
[Ausgabe von tar]
$ ^tv^x
tar -xzf /ftp/pub/Incoming/das.ist.neu.tar.gz
$ _
```

Hier werden die `tar`-Optionen `-tv` durch `-x` ausgetauscht. Zuerst wird also der Inhalt des Archivs angezeigt, dann wird es im nächsten Schritt ausgepackt. Mit dieser Methode lassen sich ebenso leicht Tippfehler in der letzten Kommandozeile verbessern.

## Anpassung des Kommandozeileneditors

Der Kommandozeileneditor der `bash` benutzt die GNU-**readline** Bibliotheksfunktionen. Wie bereits gesagt, bietet `readline` zwei Standardbelegungen an, zwischen denen Sie mit dem [set-Shellkommando](#) hin und her schalten können.

Die Zuordnung von Tastenkombinationen zu bestimmten Editorfunktionen kann aber noch weiter Ihren speziellen Bedürfnissen angepaßt werden. Sie können jede Editorfunktion mit einer frei wählbaren Tastenkombination verknüpfen. Diese Um- bzw. Neubelegung der Tasten kann zur Laufzeit mit dem [bind-Shellkommando](#) durchgeführt werden.

Das Format einer Tastaturbelegung sieht folgendermaßen aus:

### ***Taste : Kommandobezeichnung***

Die Kommandobezeichnungen sind in Klammern hinter den Beschreibungen der Editorfunktionen auf den vorhergehenden Seiten angegeben.

Für die Tastenkombinationen können die Konstruktionen `\C-` für die Kombination mit `CONTROL` und `\e-` oder `\M-` für die Metakombinationen mit `ESC` oder `ALT` verwendet werden. Außerdem können die folgenden Tasten benannt werden:

`RUBOUT` für Backspace  
`DEL` für Delete  
`ESC` für Escape  
`SPACE` oder `SPC` für Leerzeichen  
`RETURN`, `RET`, `NEWLINE` oder `LFD` für Zeilenende  
`TAB` für Tabulator

Wenn Sie z. B. anstelle des aufwendigen `set`-Kommandos die Tastenkombination `CONTROL-X v` zum Umschalten in den `vi`-Modus und die Kombination `CONTROL-X e` zum Zurückschalten in den `EMACS`-Modus verwenden wollen, erreichen Sie das mit den folgenden Kommandos:


```
$ bind -m emacs '"\C-xv":vi-editing-mode'
$ bind -m vi '"\C-xe":emacs-editing-mode'
$ bind -m vi-insert '"\C-xe":emacs-editing-mode'
```

\$ \_

Neben den Editorkommandos können auch Zeichenketten und Kommandokombinationen (Makros) auf bestimmte Tasten gelegt werden.

Beispielsweise können Sie mit dem folgenden Kommando ein Makro zum Editieren der PATH-Variablen im emacs-Modus installieren:

```
$ bind -m emacs ' "\C-xp" : "PATH=${PATH}\e\C-e\C-a\ef\C-f" '
$ _
```

Es ist auch möglich, eine neue Tastaturbelegung dauerhaft einzurichten, indem die Zuordnung von Kommandos und Tastenkombinationen in einer Datei abgespeichert wird. 

Den Namen dieser Datei können Sie in der Shellvariablen INPUTRC festlegen. Wenn diese Variable nicht existiert, wird die Belegung aus der Datei ~/.inputrc gelesen.

Für jede Umbelegung muß eine Zeile in der Datei eingetragen werden. Format und Inhalt der Zeilen stimmen mit dem oben für bind beschriebenen überein.

Zusätzlich können zur Anpassung von readline einige Schalter gesetzt und Variable belegt werden. Sie werden in der Form

*set Schalter Wert*

in der .inputrc-Datei eingetragen. Die Schalter können auch mit dem bind-Kommando benutzt werden. Die Werte in Klammern stellen die Voreinstellung dar.

bell-style

(audible) regelt das Verhalten von readline, wenn dem Benutzer ein Signal gegeben werden soll. Außer audible sind none und visible möglich.

comment-begin

(#) In dieser Variablen kann das Zeichen bestimmt werden, das durch das vi-comment-Kommando in die erste Spalte der Kommandozeile geschrieben wird. Durch das Nummernzeichen (Voreinstellung) wird die Ausführung der Zeile unterdrückt.

completion-query-items

(100) In dieser Variablen kann eine Anzahl bestimmt werden, bis zu der mögliche Erweiterungen ohne Nachfrage angezeigt werden.

convert-meta

(On) Bleibt dieser Schalter gesetzt, werden Zeichen mit gesetztem achten Bit zu einer ESC-Sequenz mit dem entsprechenden ASCII-Zeichen verarbeitet.

disable-completion

(Off) Das Setzen dieses Schalters unterdrückt die Kommandozeilenerweiterung.

editing-mode

(emacs) Durch Belegen dieser Variablen mit „vi“ wird die vi-Tastaturbelegung eingeschaltet. Einzige Alternative ist der voreingestellte emacs-Modus.

enable-keypad

(Off) Wenn dieser Schalter auf “On” gesetzt wird, versucht readline den Ziffernblock zu



aktivieren.

#### expand-tilde

(Off) Wenn der Schalter eingeschaltet wird, führt `readline` eine Tildenerweiterung bereits beim Versuch einer Kommandozeilenkomplettierung aus.

#### horizontal-scroll-mode

(Off) Wird dieser Schalter auf „On“ gesetzt, so wird beim Schreiben der Kommandozeile über den rechten Bildschirmrand hinaus die Zeile nach links verschoben, ansonsten wird sie umgebrochen.

#### input-meta

(Off) Wenn dieser Schalter „On“ gesetzt wird, erlaubt die Shell die Eingabe von 8-Bit-Zeichen (zum Beispiel Umlaute) auf der Kommandozeile.

#### keymap

(emacs) Diese Variable stellt die Tastaturtabelle für `readline` ein. Gültige Werte sind `emacs`, `emacs-standard`, `emacs-meta`, `emacs-ctlx`, `vi`, `vi-standard`, `vi-command` oder `vi-insert`. `vi` ist äquivalent zu `vi-command` und `emacs` entspricht `emacs-standard`.

#### mark-directories

(On) Solange der Schalter auf „On“ bleibt, wird bei der Kommandozeilenerweiterung ein / an einen Verzeichnisnamen angehängt.

#### mark-modified-lines

(Off) Wird dieser Schalter „On“ gesetzt, werden alle Zeilen in der `history` mit einem Asterisk `*` gekennzeichnet, die im Editor verändert wurden.

#### meta-flag

(Off) entspricht dem Schalter `input-meta`.

#### output-meta

(Off) Wenn dieser Schalter „On“ gesetzt wird, zeigt die `bash` auf der Kommandozeile 8-Bit-Zeichen an.

#### show-all-if-ambiguous

(Off) Die Veränderung dieses Schalters auf „On“ veranlaßt `readline`, bei Komplettierungsversuchen mit mehreren Möglichkeiten sofort alle Varianten anzuzeigen, anstatt ein Signal zu geben.

#### visible-stats

(Off) Wenn dieser Schalter auf „On“ gesetzt wird, erscheinen bei der Liste möglicher Erweiterungen die Symbole für die Dateitypen, wie sie auch von `ls -F` angezeigt werden.

## Bedingte Ausführung von `.inputrc`

Ähnlich wie beim C-Präprozessor können Teile der `.inputrc`-Datei durch die Direktiven `$if`, `$else` und `$endif` eingeschlossen werden, um diese Einstellungen nur unter bestimmten Bedingungen auszuführen. Folgende Tests sind vorgesehen:

`$if term=`***Terminal***

testet auf die Übereinstimmung mit der `TERM`-Umgebungsvariablen.

`$if bash`

leitet den Teil von `.inputrc` ein, der speziell für die `bash` bestimmt ist. Andere Programme, die mit der `readline`-Library arbeiten, werden mit der selben `.inputrc`-Datei initialisiert.

`$if mode=`***Modus***

ermöglicht die Unterscheidung der verschiedenen Editiermodi: `emacs`, `emacs-meta`, `emacs-ctlx`, `vi`, `vi-move` und `vi-insert`.

## Beispiel:

```
$if Bash
    # Umlaute in der Kommandozeile erlauben:
    set convert-meta Off
    set meta-flag On
    set output-meta On
    $if term=xterm
    # Spezielle Einstellungen fuer xterm
    $else
    # Einstellungen fuer alle anderen Terminals
    $endif
    $if mode=vi
    # Tastaturbelegung im vi-Modus
    $endif
    $if mode=emacs
    # Tastaturbelegung im emacs-Modus
    $endif
$endif
```

Eine ausführliche `TEXinfo`-Beschreibung aller `readline`-Funktionen kann mit dem `info`-Kommando nachgelesen werden.

Die aktuelle Tastaturbelegung wird mit dem Kommando `'bind -v'` ausgegeben.

## Der erweiterte Editor: sekundärer Prompt

Viele Operationen der Shell werden mit bestimmten Symbolen eingeleitet und müssen mit weiteren Symbolen abgeschlossen werden. Beispielsweise muß eine `for`-Schleife durch ein `done` abgeschlossen werden, eine `(`-Klammer durch eine `)`, eine mit Anführungszeichen begonnene Zeichenkette muß mit Anführungszeichen abgeschlossen werden und so weiter.

Die Shell unterstützt die Eingabe solcher Kommandos, indem sie nach einem Zeilenende in einer unvollständigen Kommandozeile eine neue Eingabeaufforderung - in der Regel ein `>` - ausgibt, bis der erwartete Abschluß des begonnenen Kommandos eingegeben ist. Die so eingegebenen Teile werden zusammen als eine Kommandozeile interpretiert.

Wenn die Shelloption `cmdhist` gesetzt ist, wird ein aus mehreren Zeilen bestehendes Kommando als ein einziger Eintrag im History-Speicher behandelt. Vor `bash` Version 2.0 wurde dieses Feature durch die Shellvariable `command_oriented_history` gesteuert.

# Interpretation der Kommandozeile

Mit dem `accept-line`-Kommando (`RETURN`) im Editor ist der im eigentlichen Sinn interaktive Teil der Shellbenutzung zu Ende. Die folgende Interpretation der Zeile durch die Shell führt normalerweise zur Erzeugung eines neuen Prozesses, in dem das eben aufgerufene Programm abläuft. Bevor das geschieht, werden aber von der Shell selbst noch eine ganze Reihe von Veränderungen an der Zeile durchgeführt. Diese Interpretation findet bei der interaktiven Shell auf exakt die gleiche Weise statt wie im Shellprogramm.

Die Kommandozeile wird analysiert, indem sie anhand von „Trennzeichen“ in atomare Sinneinheiten (Wörter oder Token) geteilt wird. Dabei kann die gesamte Zeile in mehrere Kommandos zerfallen, die getrennt weiterverarbeitet werden.

Die Wörter der einzelnen Kommandos werden sequentiell (der Reihe nach) auf bestimmte Symbole oder Sonderzeichen durchsucht. Bestimmte Sonderzeichen führen zur Umlenkung der Ein- und Ausgabekanäle. Durch andere Symbole werden bestimmte Wörter verändert oder ersetzt. Dieser Vorgang wird als Parametersubstitution bezeichnet. Danach wird die Kommandozeile noch „aufgeräumt“. Abschließend werden die Programme (intern oder extern) lokalisiert und mit ihren Optionen und Argumenten aufgerufen.

## Kommentare

In Shellscripthen können an beliebigen Stellen Kommentare durch ein Nummern- oder Hash-Zeichen ``#'` eingeleitet werden. Der gesamte Zeilenrest wird bei der Interpretation von der Shell ignoriert.

Im Unterschied zur Bourne-Shell bietet die interaktive `bash` diese Möglichkeit normalerweise nicht auf der Kommandozeile (wohl aber beim `source`-Shellkommando). Durch das Kommando `set -o interactive-comments` kann die Verwendung von Kommentaren in der interaktiven Shell ermöglicht werden.

Auf der Kommandozeile (wie auch im Shellscrip) werden Zeilen, die mit einem Doppelpunkt beginnen, zwar der Parametererweiterung unterworfen, sie werden aber nicht ausgeführt.

Ein häufiger Spezialfall tritt auf, wenn auf ein Kommentarzeichen in der ersten Zeile ein Ausrufezeichen folgt und auf dem Rest der ersten Zeile ein Interpreterprogramm benannt ist. Wenn so eine Textdatei ausführbar ist und anstelle eines Kommandos aufgerufen wird, übergibt der Linux-Kernel die Textdatei direkt dem benannten Interpreterprogramm, ohne extra eine Shell aufzurufen. Natürlich kann auch eine Shell als Interpreterprogramm für ein Shellscrip benannt werden. Das folgende Script ersetzt beispielsweise die fehlende `pwd`-Funktion der `tcsh`:

```
#!/bin/bash
IFS=" "
builtin pwd
```

## Der Status

Jede Funktion und jedes Kommando kann eine einzige Zahl an die aufrufende Shell oder allgemeiner

an das aufrufende Programm zurückgeben. Dieser Rückgabewert wird als **Status** bezeichnet.

Anders als z.B. in C-Funktionen gilt in der `bash` (wie auch in den anderen Shells) folgende Konvention:

- Der Status Null bedeutet, daß bei der Programmausführung kein Fehler aufgetreten ist. Dieser Status wird auch als „wahr“ interpretiert.
- Jeder von Null verschiedene Status wird in Shellprogrammen als „falsch“ interpretiert. Ob für so einen Status tatsächlich ein Fehler bei der Ausführung des Kommandos verantwortlich ist, hängt von dem speziellen Kommando ab.

In der Shell kann der Status des zuletzt ausgeführten Kommandos aus der Shellvariablen `?` gelesen werden.

Häufig kann aus dem Statuswert auf die Art des Fehlers zurückgeschlossen werden. Wenn ein Kommando beispielsweise durch ein Signal beendet wurde, ist der Rückgabewert (Status) der Wert des Signals + 128.

Der Status kann in sein logisches Gegenteil verkehrt werden (0 oder 1), indem dem gesamten Kommando ein ``!` vorangestellt wird. Um diese Konstruktion von der History-Substitution im C-Shell-Stil zu unterscheiden, muß dem Ausrufezeichen unmittelbar ein Blank oder ``(` folgen.

# Shell Grammatik

## Reservierte Wörter

Wie bei jeder anderen Programmiersprache gibt es für die Sprache zur Shellprogrammierung reservierte Wörter. Diese Wörter dürfen nicht für Variablennamen oder zum Benennen von Scriptfunktionen benutzt werden. Sie werden nur erkannt, wenn sie ohne Anführungszeichen und als erstes Wort eines einfachen Kommandos oder als drittes Wort eines `case`- oder `for`-Kommandos auftreten.

Folgende Wörter sind reserviert:

```
! case do done elif else esac fi for function if in select then time
until while { }
```

## Atome, Wörter, Token

Die „Atome“ einer Kommandozeile - auch als Wörter oder Token bezeichnet - werden anhand bestimmter Trennzeichen identifiziert. Die einfachste und natürliche Trennung zweier Wörter findet durch Leerzeichen statt. Dabei ist die Anzahl der Trennzeichen gleichgültig. Anstelle von Leerzeichen können auch `TABS` benutzt werden. Wegen ihrer offenkundigen Ähnlichkeit werden diese Trenner auch als **Blanks** bezeichnet.

Die folgenden zur Shellprogrammierung verwendeten **Sonderzeichen** führen automatisch auch zur Trennung von Wörtern:

| & ; ( ) < > und das Zeilenende (RETURN)

Diese Zeichen haben in jedem Fall spezielle Aufgaben und können deshalb nicht einfach innerhalb von Argumenten an ein Kommando übergeben werden. Eine Reihe weiterer Zeichen werden nur in bestimmten Situationen als Operatoren erkannt. In diesen Fällen können auch sie nicht innerhalb von Argumenten an ein Kommando weitergegeben werden. Besondere Aufmerksamkeit ist bei den folgenden Zeichen geboten:

! \* ? \$ ` ' { } [ ] ^ = # " \

Wenn eines der oben genannten (Sonder-) Zeichen als Argument an ein Kommando übergeben werden soll, muß es für die Shell „entwertet“ werden.

## Kommandos - nicht unbedingt einfach

In einem Multitasking-Betriebssystem macht es durchaus Sinn, mehr als ein Kommando in einer Zeile aufzurufen. So ein zusammengesetztes Kommando besteht aus mehreren einfachen Kommandos. Zur Trennung von einfachen Kommandos werden Kontrolloperatoren verwendet, die aus den oben genannten Sonderzeichen zusammengesetzt sind:

| & || && ; ;; ( )

Außer mit dem eigentlichen Kommandonamen (dem Namen der ausführbaren Datei) kann ein einfaches Kommando auch mit einer Zuweisung an eine Shellvariable beginnen.

Für jedes einfache Kommando können die offenen Datenkanäle umgeleitet, sowie neue erzeugt werden.

## Quotierung

Quotierung wird benutzt, um die spezielle Bedeutung von Kontrollzeichen, reservierten Wörtern oder Namen auszuschalten. Auf diese Weise können Parameter an Funktionen übergeben werden, die Sonderzeichen, Namen oder reservierte Wörter enthalten, ohne daß die Shell eine Veränderung an den Parametern vornimmt.

Es gibt drei Formen der Quotierung:

1. durch das Fluchtsymbol \ (Backslash)
2. durch Hochkomma ' (Quote)
3. durch Anführungszeichen " (Doublequote)

Das Fluchtsymbol „entwertet“ das unmittelbar folgende Sonderzeichen. Ein durch das Fluchtsymbol entwertetes Zeilenende wird ignoriert.

Die in Hochkommata eingeschlossenen Wörter werden von der Shell nicht weiter bearbeitet. Lediglich ein Hochkomma darf nicht in Hochkommata eingeschlossen werden; auch nicht wenn, es durch ein Fluchtsymbol eingeleitet wird.

Von den in Anführungszeichen eingeschlossenen Wörtern erkennt die Shell nur die Sonderzeichen \$, ' und \ als solche. Alle anderen Wörter bleiben unbearbeitet. Das Fluchtsymbol behält seine Bedeutung aber nur, wenn es von einem der Zeichen \$ ' " \ oder dem Zeilenende (RETURN) gefolgt wird. Ein

Anführungszeichen darf zwischen zwei Anführungszeichen stehen, wenn es durch ein Fluchtsymbol eingeleitet wird.

Die speziellen Parameter \* und @ haben eine besondere Bedeutung, wenn sie zwischen [Anführungszeichen](#) auftauchen.

Von Anführungszeichen eingeschlossene Zeichenketten, denen ein Dollarzeichen vorangestellt ist, werden nach Möglichkeit dem aktuellen Locale entsprechend übersetzt. Wenn das aktuelle Locale C oder POSIX ist, wird das Dollarzeichen ignoriert. Die resultierende Zeichenkette bleibt in Anführungszeichen eingeschlossen.

## Ein-/Ausgabe-Umleitung

Jedes Programm erhält automatisch beim Start drei offene „Datenkanäle“: die Standardeingabe, die Standardausgabe und die Standardfehlerausgabe.

Bevor eine Kommandozeile ausgeführt wird, können die bestehenden Eingabe- und Ausgabekanäle umgelenkt, sowie neue erzeugt werden. Auf diese Weise können Dateien im Dateisystem zum Lesen bzw. Schreiben für das Kommando geöffnet werden, die nach dessen Beendigung automatisch wieder geschlossen werden. Beispielsweise kann so die Ausgabe des `ls`-Kommandos zur weiteren Bearbeitung in eine Datei geschrieben werden.

Wenn mehrere Kanalumlenkungen in einer Kommandozeile auftauchen, werden sie der Reihe nach von links nach rechts ausgewertet.

Wenn in einer der folgenden Beschreibungen die Kanalnummer einer Datei nicht angegeben wird, so wird bei einer Eingabeumleitung die Standardeingabe (Kanal 0) und bei einer Ausgabeumleitung die Standardausgabe (Kanal 1) umgeleitet.

Das auf den Umleitungsoperator folgende Wort wird allen möglichen Parametererweiterungen unterworfen. Wenn durch die Erweiterung mehr als ein Wort entsteht, wird eine Fehlermeldung ausgegeben.

Mit Hilfe des `exec`-Shellkommandos kann auch die Eingabe/Ausgabe der aktiven Shell umgelenkt werden, indem `exec` ohne Kommando, aber mit entsprechenden Umleitungen aufgerufen wird ([siehe exec](#)).

Wenn in einem Kommando mehrere Umleitungen gelegt werden, ist die Reihenfolge signifikant. Ein Beispiel hierfür ist bei der [Verdoppelung der Dateikennung](#) gegeben.

## Eingabeumleitung

**[*n*]<Wort**

erzeugt entweder eine Eingabeumleitung von der mit dem (erweiterten) Wort bezeichneten Datei auf den Kanal mit der Nummer *n*, oder auf die Standardeingabe (Kanal 0), wenn keine Zahl *n* angegeben wird.

# Ausgabeumleitung

**[n]>Wort**

lenkt den Ausgabekanal mit der Nummer *n* auf die mit dem Wort bezeichnete Datei um. Wenn keine Zahl für die Kanalnummer angegeben ist, wird die Standardausgabe (Kanal 1) angenommen.

Wenn die angegebene Datei nicht existiert, wird sie erzeugt. Wenn eine Datei dieses Namens existiert, wird sie überschrieben, solange die Shelloption [noclobber](#) nicht gesetzt ist.

## Anfügen der Ausgabe an eine existierende Datei

**[n]>>Wort**

öffnet die Datei mit dem angegebenen Namen zum Anhängen von Daten. Die Daten aus dem Ausgabekanal mit der Nummer *n* werden an die Datei angehängt. Wenn die Kanalnummer fehlt, wird die Standardausgabe umgelenkt.

Wenn die Datei nicht existiert, wird sie erzeugt.

## Zusammenfassung der Standardausgabe mit der Standardfehlerausgabe

**&>Wort** oder  
**>&Wort**

legt die Kanäle für die Standardausgabe und die Standardfehlerausgabe zusammen und schreibt sie in eine Datei namens *Wort*. Von den beiden Formen sollte die erste bevorzugt werden.

## Shellscript-Dokumente

**<<[-]Wort**  
**Dokument**  
**Begrenzer**

Diese Art der Kanalumlenkung wird benutzt, um einen Teil des Shellscripts direkt als Standardeingabe für ein Kommando zu benutzen. Es werden alle Zeilen des Dokumentes gelesen und als Eingabe an das Kommando weitergeleitet, bis eine Zeile auftaucht, die nur den Begrenzer enthält. Das in der ersten Zeile festgelegte Wort wird keiner Erweiterung unterzogen. Wenn es aber irgendeine Art der Quotierung enthält, ist der Begrenzer das Wort ohne die Quotierung. In diesem Fall wird aber das Dokument ohne jede Erweiterung an das Kommando weitergegeben. Anderenfalls (wenn das Wort keine Quotierung enthält) werden alle Parameter im Dokumenttext erweitert und es wird die Kommandosubstitution durchgeführt.

In dem zweiten Fall wird ein durch das Fluchtsymbol ``` eingeleitetes Zeilenende ignoriert. Um das Fluchtsymbol selbst sowie die Zeichen `$` und `'` im Dokument darstellen zu können, müssen sie ebenfalls durch ein Fluchtsymbol eingeleitet werden.

Wenn das optionale Minuszeichen bei der Einleitung des Shellsript-Dokumentes auftaucht, werden alle Leerzeichen und Tabulatoren am Anfang der Dokumentzeilen ignoriert. Dadurch kann das Dokument durch Einrückung deutlich von dem übrigen Shellsript abgesetzt werden, ohne daß diese Einrückung auch in der Ausgabe erscheint.

## Verdoppelung der Dateikennung

**[*n*]<&*Wort***

kopiert die in *Wort* enthaltene (ganzahlige) Eingabedateikennung. Es wird die neue Dateikennung *n* als Eingabekanal erzeugt oder eine existierende Kennung überschrieben.

Wenn im *Wort* anstelle einer Zahl ein '-' steht, wird der Kanal *n* geschlossen. Wenn das Wort leer ist, wird es durch die Standardeingabe ersetzt.

**[*n*]>&*Wort***

verdoppelt die Ausgabedateikennung in *Wort*. Wenn das Wort leer ist, wird hier die Standardausgabe eingesetzt. Ansonsten ist die Funktion die gleiche wie bei der Verdoppelung der Eingabekennung.

Ein **Beispiel** soll die Funktion der Verdoppelung verdeutlichen: Die Konstruktion

```
$ ls /usr/local/foo > inhalt 2>&1
$ cat inhalt
ls: /usr/local/foo: No such file or directory
$ _
```

lenkt die Standardausgabe und die Standardfehlerausgabe in die Datei `inhalt` um.

Die Konstruktion

```
$ ls /usr/local/foo 2>&1 > inhalt
ls: /usr/local/foo: No such file or directory
$ cat inhalt
$ _
```

lenkt dagegen nur die Standardausgabe in die Datei `inhalt` um.

Beim ersten Beispiel wird zuerst die Standardausgabe in die Datei umgelenkt, danach wird der Standardausgabekanal verdoppelt und dabei die Standardfehlerausgabe ersetzt. Im zweiten Beispiel wird zuerst die Standardfehlerausgabe durch die Standardausgabe ersetzt. Das ist in diesem Moment aber noch der Bildschirm. Erst danach wird die Standardausgabe mit der Datei verbunden. Die Kopie des Standardausgabekanals (die Standardfehlerausgabe) wird von dieser Umlenkung aber nicht betroffen!

## Öffnen einer Datei zum Lesen und Schreiben

**[*n*]<>*Wort***

öffnet die im *Wort* benannte Datei zum Lesen und Schreiben.



# Pipelines

Die engste Verknüpfung mehrerer einfacher Kommandos wird durch eine sogenannte Pipeline hergestellt. Das Charakteristikum einer Pipeline ist die Zusammenlegung des Standardausgabekanals des einen Kommandos mit dem Standardeingabekanal des zweiten. Dabei wird die Multitasking-Fähigkeit von Linux ausgenutzt und die beiden (oder mehr) Kommandos als separate Prozesse gleichzeitig gestartet.

Die Syntax für die Zusammenfassung zweier Kommandos in einer Pipeline sieht folgendermaßen aus:

```
[time [-p]] [ ! ] Kommando | Kommando [ | Kommando ] ...
```

Der entscheidende Kontrolloperator ist das Zeichen `|' (Pipe).

Seit der Version 2.0 benutzt die `bash` das zusätzliche Schlüsselwort `time`, das dazu dient, die Ausführungszeit der Pipeline zu ermitteln.

Der Status der gesamten Pipeline ist gleich dem Status des letzten Kommandos in der Pipeline ([siehe status](#)). Durch das Ausrufezeichen wird der Wahrheitswert des Status umgekehrt.

Die Verknüpfung der Standardkanäle findet vor einer eventuellen Umlenkung durch eines der Kommandos statt. Das bedeutet, daß die Zusammenlegung von Standardfehlerausgabe und Standardausgabe tatsächlich in der Pipeline mündet.

Die parallele Ausführung aller einfachen Kommandos einer Pipeline legt es nahe, die komplette Pipeline syntaktisch als Einheit zu betrachten. Wenn im folgenden nicht ausdrücklich von einfachen Kommandos die Rede ist, steht „Kommando“ auch für Pipelines.

Es können mehrere Kommandos durch Pipelines verkettet werden. Dabei sind aber nur zwei Pipelines für jedes einfache Kommando erlaubt - jeweils eine für die Standardeingabe und die Standardausgabe. Soll ein Kommando aus mehreren Pipelines gleichzeitig lesen, kann das ab `bash-1.13` durch die [Prozeßsubstitution](#) mit Named-Pipes realisiert werden.

## Hintergrundprozesse

Nachdem die Kommandozeile von der Shell bearbeitet ist, werden die darin enthaltenen Kommandos ausgeführt. Ein Kommando oder eine Pipeline wird dann zu einem Job. Dieser Job läuft weitgehend unabhängig von der Shell; trotzdem steht die Shell weiterhin mit allen Jobs in Verbindung und kann über Signale mit ihnen kommunizieren. Zu diesem Zweck unterhält die `bash` eine Tabelle aller aktuellen Jobs.

Wenn ein Job mit dem `&'-Zeichen im Hintergrund gestartet wird, gibt die `bash` eine Zeile mit der Jobnummer und der Prozeßnummer für diesen Job aus. Wenn ein Hintergrundjob beendet ist, wird wieder eine Meldung mit dem Namen, der Jobnummer und dem Status des Jobs ausgegeben.

Mit der Tastenkombination `^Z` ist es jederzeit möglich, einen im Vordergrund laufenden Job anzuhalten. Es erscheint dann eine Meldung mit der Jobnummer und dem Namen des angehaltenen Jobs. Danach erscheint die Eingabeaufforderung der Shell.

Ein angehaltener Job kann mit dem `bg`- oder dem `fg`- Shellkommando im Hintergrund oder im Vordergrund gestartet werden. Mit dem `kill`-Kommando wird er abgebrochen ([siehe bg, fg und kill](#)).

Im Zusammenhang mit den genannten Kommandos kann ein Job mit seiner Jobspezifikation und seiner Prozeßnummer angesprochen werden.

Als **Jobspezifikation** werden die Jobnummer oder der Anfang des Jobnamens erkannt, indem sie durch ein Prozentzeichen ``%'` eingeleitet werden. Die Jobnummer ist die laufende Nummer, unter der der Job in der Jobtabelle geführt wird. Der zuletzt angehaltene Job wird auch als aktueller Job bezeichnet und kann mit ``%+'` angesprochen werden. Der zuvorletzt angehaltene Job wird auch als der letzte Job bezeichnet und kann mit ``%-'` benannt werden. Wenn ein angegebener Anfang auf mehrere Jobs paßt, wird eine Fehlermeldung ausgegeben. Ein Job kann auch einfach durch Angabe seiner Jobspezifikation (ohne Kommando) aus dem Hintergrund in den Vordergrund geholt werden. Zum Beispiel bringt ``fg %1'` den angehaltenen oder im Hintergrund laufenden Prozeß mit der Jobnummer 1 im Vordergrund zum Laufen.

Bei der Ausgabe der Jobtabelleneinträge wird der aktuelle Job mit einem ``+'` gekennzeichnet und der letzte Job mit einem ``-'`.

Wenn die Shell beendet werden soll, während sich angehaltene Jobs im Hintergrund befinden, wird eine Warnung ausgegeben und die Shell nicht beendet. Erst wenn die Shell entweder unmittelbar darauf, oder nach einem einzigen `jobs`-Shellkommando ein zweites Mal beendet wird, werden alle verbleibenden Jobs automatisch terminiert.

Wenn zum Zeitpunkt des Ausloggens noch Jobs im Hintergrund laufen, werden diese Jobs beim Verlassen der Shell automatisch an den `init`-Prozeß übereignet und laufen so weiter.

## Listen

Neben der Möglichkeit, zwei oder mehr einfache Kommandos in einer Pipeline parallel aufzurufen, erlaubt die Shell weitere Aufrufe mehrerer Kommandos in einer einzigen Zeile. Diese Aufrufe werden als Listen bezeichnet.

Eine Folge von Kommandos kann durch die Kontrolloperatoren `&&`, `||`, `&` oder `;` zu einer Liste zusammengefaßt werden. Die Liste wird durch ein `&`, `;` oder ein Zeilenende (`RETURN`) abgeschlossen.

### Listen mit `;` und `&`

Ein Semikolon kann zum Abschluß eines Kommandos benutzt werden. Auf diese Weise können beliebige Kommandos zu einer Liste zusammengefügt werden, indem sie, durch Semikolon getrennt, auf einer Zeile eingegeben werden. Die Ausführung der beiden Kommandos geschieht unabhängig, nacheinander.

Mit dem `&`-Operator wird ebenfalls ein Kommando abgeschlossen. Auch mit diesem Operator können

also zwei Kommandos auf einer Zeile getrennt werden. Jedes mit einem & abgeschlossene Kommando wird unabhängig von den anderen Kommandos der gleichen Zeile im Hintergrund ausgeführt. Es können auf diese Weise also zwei oder mehr Kommandos parallel aufgerufen werden. Diese Kommandos stehen während der Ausführung nicht in Verbindung zueinander. Wenn alle Kommandos einer Zeile durch ein & abgeschlossen werden, erscheint bei der interaktiven Shell sofort die nächste Eingabeaufforderung.

## Bedingte Ausführung

Die Kontrolloperatoren && und || verknüpfen zwei Kommandos logisch miteinander. Das zweite Kommando dieser Liste wird in Abhängigkeit vom Ergebnis (Status) des ersten ausgeführt.


Bei der Verknüpfung durch

***Kommando1*** [ && ***Kommando2*** ...]

wird das *Kommando2* nur dann ausgeführt, wenn *Kommando1* fehlerfrei abgeschlossen wurde. Man kann das als eine logische UND-Verknüpfung der beiden Kommandos betrachten. Der Status der UND-Liste ist wahr (Null), wenn beide Kommandos „wahr“ sind. Wenn bereits das erste Kommando Null liefert, wird das zweite zur Bewertung des Gesamtausdrucks nicht mehr benötigt, also wird es auch nicht ausgeführt.

Das Pendant zur UND-Liste ist die ODER-Verknüpfung durch

***Kommando1*** [ ||***Kommando2*** ...]

Hier wird das *Kommando2* nur dann ausgeführt, wenn bei der Bearbeitung vom *Kommando1* ein Fehler aufgetreten ist. Analog zur UND-Verknüpfung kann auch die Verkettung durch || als logische ODER-Verknüpfung betrachtet werden. Der Status der ODER-Liste ist wahr, wenn das erste oder das zweite Kommando einen Status Null liefert. Wenn das bereits beim ersten Kommando erfüllt ist, wird das zweite nicht mehr ausgeführt. 

## Gruppen und Kontrollstrukturen: Blöcke, Schleifen, Verzweigungen, Funktionen

Einzelne Kommandos oder Listen können auf verschiedene Weisen weiter zu Gruppen zusammengefaßt werden:

***{Liste ;}***

Mit den geschweiften Klammern werden die Kommandos der Liste zu einer einfachen Gruppe zusammengefaßt. Diese Klammerung entspricht der in mathematischen Ausdrücken und dient der Assoziierung nieder- oder gleichrangig verknüpfter Kommandos.

Die Operatoren zur Shellprogrammierung unterscheiden sich in Rang und Assoziativität. Prinzipiell wird eine Kommandozeile von links nach rechts abgearbeitet. Die Zusammenfassung in UND- oder ODER-Listen hat gegenüber der Auflistung mit Semikolon oder & Vorrang. Die Zusammenfassung einfacher Kommandos in Pipelines hat wiederum

größere Assoziativität als die Listen. Die beiden Zeilen

```
{ cat /etc/gettydefs || cat /etc/gettytab; } | grep 38400
cat /etc/gettydefs || cat /etc/gettytab | grep 38400
```

unterscheiden sich syntaktisch nur in der Klammerung, die größere Assoziativität der Pipeline führt in der zweiten Zeile aber zu einer impliziten Klammerung der beiden letzten Kommandos. Die erste Zeile schreibt die erste existierende der beiden angegebenen Dateien in den Standardausgabekanal, der durch eine Pipeline dem grep-Kommando zur Auswertung übergeben wird. Die zweite Zeile schreibt entweder die Datei /etc/gettydefs auf den Bildschirm (Standardausgabe), oder sie übergibt die Datei /etc/gettytab dem grep-Kommando.

Die geschweiften Klammern trennen selbst keine Kommandos (siehe oben). Aus diesem Grund ist ein Leerzeichen zwischen der einleitenden Klammer und dem ersten Kommando der eingeschlossenen Liste notwendig. Um die Trennung zum ersten Kommando nach der abschließenden Klammer eindeutig anzugeben, ist es sinnvoll, die eingeklammerte Liste immer mit einem Semikolon abzuschließen. Im Gegensatz zur Bourne-Shell wird in dem Beispiel oben das Kommando korrekt interpretiert, wenn das Semikolon weggelassen wird, weil die der Klammer folgende Pipeline die Kommandos trennt.

Im Unterschied zur Bourne-Shell wird bei der bash auch in dem Beispiel oder bei einer Ausgabeumlenkung für die komplette Klammer keine Subshell erzeugt.

## (Liste)

Durch runde Klammerung werden die Kommandos einer *Liste* in einer eigenen (Shell-) Umgebung ausgeführt. Wenn von den Kommandos der Liste Veränderungen an der Shellumgebung vorgenommen werden, sind diese außerhalb der eingeklammerten Gruppe nicht sichtbar. Wenn zum Beispiel innerhalb einer so geklammerten Gruppe das Verzeichnis gewechselt wird, beziehen sich alle weiteren Kommandos innerhalb der Gruppe auf dieses Verzeichnis; außerhalb der Klammerung bleibt das „alte“ Verzeichnis aktuell. Beispiel:

```
cd /
$ (cd /usr/bin; ls e*); ls -F
egrep      elvis      elvprsv     elvrec      env          ex          expand
bin/        etc/              lost+found/  root/        usr/
boot/       home/            mnt/         sbin/        var/
dev/        lib/             proc/         tmp/         vmlinuz
$ _
```

erzeugt eine Subshell, in der aus dem aktuellen Wurzelverzeichnis in das Verzeichnis /usr/bin gewechselt wird. Hier können beispielsweise alle Dateien angezeigt werden, deren Name mit einem `e' beginnt. Mit den schließenden Klammern wird auch die Subshell beendet. Das darauffolgende Listing findet wieder im ursprünglich aktuellen Verzeichnis statt.

## for Name [in Wort] do Liste done

Mit dieser Konstruktion stellt die bash die von vielen Programmiersprachen bekannte und bewährte for-Schleife bereit. Mit *Name* wird eine Shellvariable definiert, die in jedem Schleifendurchlauf einen neuen Wert erhält. Die Anzahl der Schleifendurchläufe entspricht der Anzahl der vorhandenen Werte.

Die Werte werden normalerweise nach dem Schlüsselwort *in* übergeben. Dazu können mehrere *Wörter* angegeben werden, die von der Shell [erweitert werden](#). Für jeden Durchlauf wird ein Token in der Variablen *Name* übergeben.

Wenn der ``in Wort'` Teil fehlt, wird die Liste für jeden gesetzten [Positionsparameter](#) einmal ausgeführt.

Die interaktive Eingabe einer `for`-Schleife wird (wie die Eingabe jedes anderen aus mehreren Teilen bestehenden Konstrukts) von der Shell unterstützt, indem sie einen sekundären Prompt (PS2) zur Eingabe einer Schleife über mehrere Zeilen anbietet. Dieser sekundäre Prompt wird ausgegeben, solange die Schleife nicht mit `done` abgeschlossen ist.

Als Status wird der Rückgabewert des letzten ausgeführten Kommandos zurückgegeben. Wenn kein Kommando ausgeführt wurde, ist der Status Null.

In dem folgenden Beispiel werden im aktuellen Verzeichnis alle Dateien mit der Endung ``foo'` umbenannt, so daß sie auf ``bar'` enden.

```
$ for i in *.foo; do
> base=`basename $i .foo`
> mv $i $base.bar
> done
$ _
```

Die Zuweisung der Ausgabe vom Shellutility `basename` an die Variable `base` findet durch [„Kommandosubstitution“](#) statt. Diese Operation wird durch die „Backquotes“ ausgelöst, nicht durch Hochkomma.

Die `bash` bietet mit ihrer Parametererweiterung eine andere schöne Methode, die Endung vom Dateinamen zu trennen: mit der Konstruktion `base=${i%.foo}` ([Parametererweiterung](#)). Das spart den Aufruf eines externen Kommandos und verbessert damit die Performance.

Wenn eine Variable innerhalb einer `for`-Schleife verändert wird, ist der neue Wert auch außerhalb der Schleife sichtbar. Die `bash` verhält sich damit POSIX-1003.2-konform. Eine Ausnahme kann für die Zählvariable erreicht werden, indem das Kommando `set -l` gegeben wird. In diesem Fall wird die Zählvariable nach dem letzten Durchlauf auf den Wert vor Beginn der Schleife zurückgesetzt.

## **while Liste do Liste done**

Von anderen Programmiersprachen ebenso bekannt ist die `while ... do ... done`-Schleife. Hier wird der Schleifenkörper ``do Liste done'` so lange wiederholt, bis die in ``while Liste'` formulierte Bedingung falsch ist. Das ist der Fall, wenn das letzte Kommando der `while Liste` einen Status ungleich Null liefert.

Bei der interaktiven Eingabe einer `while`-Schleife wird so lange der sekundäre Prompt (PS2) ausgegeben, bis die Schleife mit `done` abgeschlossen ist.

Der Status der `while`-Schleife ist gleich dem Status des letzten Kommandos des ``do-Teils'` oder Null, wenn kein Kommando ausgeführt wurde.

Im folgenden Beispiel wird die `while`-Schleife benutzt, um die Unterverzeichnisse z. B. des Verzeichnisses `/usr/local/man` anzulegen.

```
$ declare -i zahl=1
$ while [ $zahl -lt 10 ]
> do
> mkdir man$zahl
> mkdir cat$zahl
```

```
> zahl=$((zahl+1))
> done
$ _
```

Die Deklaration von `zahl` als Integer-Variable ist an dieser Stelle nicht notwendig, weil die Zuweisung einer Zahl und die Verwendung in arithmetischen Ausdrücken diese Interpretation implizieren.

Zur Inkrementierung der Zählvariablen wird die Arithmetische Erweiterung benutzt, die [hier](#) beschrieben ist. Die `bash` erlaubt ab Version 1.13 auch die direkte Inkrementierung einer Integer-Variablen durch den `+=` Zuweisungsoperator in einer arithmetischen Erweiterung oder mit dem `let`-Shellkommando. Beide Varianten sind in der Standard-Bourne-Shell nicht vorgesehen.

### **`until Liste do Liste done`**

Die `until`-Schleife entspricht der `while`-Schleife mit dem Unterschied, daß der `do`-Teil so lange ausgeführt wird, wie das letzte Kommando der `until Liste` einen Status ungleich Null liefert.

### **`if Liste then Liste [elif Liste then Liste ...] [else Liste] fi`**

Mit der `if`-Konstruktion werden die Kommandos der `then Liste` unter der Bedingung ausgeführt, daß das letzte Kommando der `if Liste` „wahr“ ist, also einen Status Null liefert.

Mit der optionalen `elif`-Konstruktion können beliebig lange `if-else`-Ketten erzeugt werden. Wenn der `elif Liste`-Teil des letzten `elif` nicht Null liefert, wird der abschließende `else`-Teil bearbeitet.

Wie bei den Schleifen wird auch bei der interaktiven Eingabe einer `if`-Anweisung der sekundäre Prompt ausgegeben, bis die Konstruktion mit einem `fi` abgeschlossen ist.

Der Status der gesamten `if`-Konstruktion ist gleich dem Status des zuletzt ausgeführten Kommandos, oder Null, wenn kein Kommando ausgeführt wurde.

Das folgende Beispiel zeigt, wie in der Initialisierungsdatei `~/ .bashrc` zwischen einer Shell im `xterm` und den Textbildschirmen unterschieden werden kann.

```
if [ $WINDOWID ]; then
    TERM=xterm
    export XDVIFONTS=/usr/TeX/lib/tex/fonts/%f.%d%p
    export OPENWINHOME=/usr/openwin
    export PAGER=/usr/X386/bin/xless
else
    export PAGER=/usr/bin/less
fi
```

Ein weiteres Beispiel zeigt, wie in einem Shellscript unterschieden werden kann, ob die aufrufende Shell interaktiv arbeitet oder nicht.

```
if [ "${-#*i}" = "$-" ]; then
    echo Die Shell arbeitet nicht interaktiv
else
    echo Die Shell arbeitet interaktiv
fi
```

Der in diesem Beispiel benutzte Ausdruck `[ "${-#*i}" = "$-" ]` ist ein schönes Beispiel für die Parametererweiterung, wie sie [hier](#) erklärt ist. Im Spezialparameter `-` sind die Optionsflags der Shell



gespeichert. Durch die „Erweiterung“ `$ { -#*i }` wird aus dieser Zeichenkette ein ``i'` (und alle Zeichen davor) entfernt, wenn es enthalten ist. Der umschließende [test](#) vergleicht die so eventuell verkürzte Zeichenkette mit dem Original. Wenn sie gleich sind, ist die ``i'`-Option der interaktiven Shell nicht gesetzt.

Die Anführungszeichen sind in dem Beispiel notwendig, weil die Shellvariable ``-'` auch leer sein kann. In diesem Fall würde sie ohne die Anführungszeichen aus der Kommandozeile entfernt, was in dem Vergleich zu einem verdeckten Syntaxfehler und einem falschen Ergebnis des Tests führen könnte.

**case Wort in [Muster [ |Muster ...] ) Liste ;; ...] esac**

Mit der `case`-Anweisung können leicht Verzweigungen programmiert werden, bei denen viele Fälle unterschieden werden.

Das **case Wort** wird von der `bash` erweitert, dann wird die daraus entstandene Zeichenkette mit den Mustern verglichen und bei Übereinstimmung die Liste von Kommandos ausgeführt. In den Suchmustern können reguläre Ausdrücke wie bei der Pfadnamenerweiterung (z. B. ``*'`` und ``?'``) verwendet werden.

Wenn ein übereinstimmendes Muster gefunden wurde, wird die `case`-Anweisung beendet und nicht nach weiteren Übereinstimmungen gesucht.

Der Status ist Null, wenn keine Übereinstimmung gefunden wurde. Sonst wird der Status des zuletzt ausgeführten Kommandos der Liste übergeben.

Das dem letzten Beispiel zugrunde liegende Problem kann auch mit der `case`-Anweisung gelöst werden. Diese Lösung ist zwar kein typisches Beispiel für eine Vielfachverzweigung, sie bietet sich aber wegen der Auswertung regulärer Ausdrücke durch `case` in der Praxis eher an als das oben gegebene Beispiel.

```
case $- in
    *i\*) echo hier sollte nach der Zeichenkette 'i*'
          echo gesucht werden.
          echo      *****FEHLER***** in der bash-1.12.;;
    *i*)  echo Die Shell arbeitet interaktiv.;;
    *)    echo Die Shell arbeitet nicht interaktiv.;;
esac
```

In der `bash` Version 1.12 konnte mit der `case`-Anweisung nicht nach regulären Ausdrücken selbst gesucht werden, die Quotierung der Wildcards wurde ignoriert. Ab Version 1.13 ist dieser Fehler behoben.

**select Name [in Wort...;] do Liste done**

Die `select`-Kontrollstruktur bietet eine Kombination aus menügesteuerter Verzweigung und Schleife.

Der `in Wort`-Teil wird erweitert und die so generierten Wörter als numerierte Liste (Menü) auf dem Standardfehlerkanal ausgegeben. Mit dem PS3-Prompt wird daraufhin eine Eingabe von der Tastatur gelesen. Eine leere Eingabe führt zu einer erneuten Anzeige des Menüs.

Wenn ein Wort aus der Liste durch seine Nummer bestimmt wird, führt die `bash` die Kommandos der `do Liste` aus und stellt dabei das ausgewählte Wort in der Variablen `Name` zur Verfügung. Wird in der Eingabezeile keine passende Zahl übergeben, ist `Name` leer, die Eingabezeile ist aber in der Variablen `REPLY` gespeichert.

Menüteil und Ausführung der Liste werden so lange wiederholt, bis die Schleife mit `break` oder `return` verlassen wird. Es ist möglich, mit `CONTROL-D` das Menü unmittelbar zu

verlassen.

Wenn der *in Wort*-Teil fehlt, werden stattdessen die Positionsparameter verwendet.

**[function] Name ( ) {Liste}**

definiert eine neue Scriptfunktion *Name*.

Scriptfunktionen sind Teile eines Shellscripts oder einer Initialisierungsdatei für eine interaktive Shell, die komplett in der Shellumgebung gespeichert werden. Wenn ein Kommando in der Kommandozeile mit einer solchen Funktion übereinstimmt, wird die unter dem Funktionsnamen gespeicherte *Liste* von Kommandos ausgeführt. Es wird dazu kein neuer Prozeß erzeugt.

Im Unterschied zu *alias*-Synonymen können Scriptfunktionen Argumente verarbeiten. Wenn die Scriptfunktion mit Argumenten aufgerufen wird, werden diese Argumente der Funktion als Positionsparameter übergeben. Der Spezialparameter ``#'` wird aktualisiert. Der Positionsparameter ``0'` bleibt allerdings unverändert.

Mit dem `local`-Shellkommando ist es möglich, lokale Variablen für Scriptfunktionen zu erzeugen. Normale Shellvariablen sind „global“, sind also in der gesamten Shell uneingeschränkt sichtbar.

Wenn in einer Scriptfunktion das Shellkommando `return` auftaucht, wird die Funktion beendet und mit der dem Aufruf folgenden Zeile des Shellscripts oder der interaktiven Eingabe fortgesetzt. Die Positionsparameter und der Spezialparameter ``#'` werden auf ihre Werte vor dem Funktionsaufruf zurückgesetzt.

Eine Liste der definierten Scriptfunktionen erhält man in einer interaktiven Shell mit der Option ``-f'` zu den Shellkommandos `declare` oder `typeset`. Die Scriptfunktionen werden nur dann an alle Unterschells weitergereicht (und stehen nur dann auch in diesen Shells zur Verfügung), wenn sie mit der Shellfunktion `export` unter der Option ``-f'` für den Export bestimmt wurden.

Scriptfunktionen können rekursiv aufgerufen werden. Es gibt keine Begrenzung für die Anzahl der rekursiven Aufrufe.

## Parameter

Aus dem Blickwinkel des Shellprogrammierers ist die zentrale Rolle von Variablen unmittelbar klar. Variable sind symbolische Namen für Platzhalter, in denen Werte (Zahlen oder Zeichenketten) gespeichert werden können.

Eine Variable kann durch eine Zuweisung der folgenden Form erzeugt werden:

***Name*=[Wert]**

Als Namen für Variable kommen beliebige alphanumerische (ASCII-) Zeichenketten in Frage, die mit einem Buchstaben beginnen. Der Unterstrich ``_'` wird zu den erlaubten Buchstaben gerechnet.

Zwischen den Bestandteilen der Zuweisung dürfen keine Leerzeichen stehen.

Wenn kein Wert angegeben ist, wird der Variablen die leere Zeichenkette (Nullstring) zugewiesen.

Es ist nicht notwendig (aber möglich), Variable zu deklarieren. Automatisch erzeugte Variable speichern alle ihnen zugewiesenen Werte als Zeichenketten. Die Interpretation des Inhalts einer Variablen erfolgt „aus dem Zusammenhang“. Das heißt, in arithmetischen Ausdrücken werden Ziffernfolgen automatisch als Zahlen interpretiert. Durch ausdrückliche Deklaration einer Variablen als



Integer (mit der Shellfunktion `declare -i`) wird erreicht, daß jede Zuweisung automatisch der arithmetischen Erweiterung unterworfen wird. Damit ist garantiert, daß eine solche Variable nach ihrer Initialisierung immer Zahlen enthält.

Der Zugriff auf den Wert eines Parameters bzw. die Übergabe eines Parameters erfolgt durch den Operator ``$'` gefolgt von der Parameterbezeichnung.

Ein Parameter gilt als gesetzt, wenn ihm ein Wert zugewiesen wurde. Eine leere Zuweisung - der Nullstring `' '` - gilt als Wert in diesem Sinne. Wenn eine Variable gesetzt ist, kann sie nur durch das `unset`-Kommando entfernt werden.

## Shell- und Umgebungsvariable

Shellvariable haben auch in der interaktiven Shell eine große Bedeutung.

- Zum einen können bestimmte Variable aus der allgemeinen Prozeßumgebung (Environment) der Shell wie Shellvariable benutzt werden. Einige Umgebungsvariablen existieren bereits, wenn die Shell aufgerufen wird. Neue Umgebungsvariable können mit dem [export-Shellkommando](#) erzeugt werden. Die Prozeßumgebung wird vom Elternprozeß an die Kinder vererbt, so daß alle Umgebungsvariablen der `bash` an die von dieser Shell gestarteten Prozesse weitergegeben werden. Die Kindprozesse können dann aus ihrer Prozeßumgebung wichtige Informationen über das System, in dem sie laufen, erhalten.
- Zum anderen benutzt die Shell selbst Variable nicht nur als Platzhalter in Shellprogrammen, sondern sie verwendet bestimmte Shellvariable zur Anpassung ihres eigenen Verhaltens an eine bestimmte Systemumgebung. Der Benutzer kann diese Variablen verändern und damit das Erscheinungsbild der Shell seinen Bedürfnissen und seinem Geschmack entsprechend einrichten.

Auf der Benutzerebene können Umgebungsvariable wie Shellvariable beliebig erzeugt, gelesen und verändert werden. Deshalb erscheinen die Umgebungsvariablen als eine Untermenge der Shellvariablen. Umgebungsvariable können mit dem `printenv`-Kommando oder dem `export`-Shellkommando angezeigt werden. Sämtliche definierten Shellvariablen werden mit dem `set`-Shellkommando ausgegeben.

Die folgenden Variablen werden von der Shell ausgewertet. In einigen Fällen wird die Variable mit einem Standardwert initialisiert.

### **BASH\_ENV**

entspricht in der Funktion der Shellvariablen `ENV`. Im gegensatz zu letzterer wird `BASH_ENV` jedoch nur verwendet, wenn die Shell zur Bearbeitung eines Shellscripts unter den Namen `bash` aufgerufen wird.

### **CDPATH**

ist eine durch Doppelpunkt getrennte Liste von Verzeichnissen. Wenn beim `cd`-Shellkommando kein absoluter Verzeichnisname und kein Verzeichnis relativ zum aktuellen Verzeichnis benannt wird, werden alle Verzeichnisse im `CDPATH` nach einem passenden Verzeichnis durchsucht. In das erste passende Verzeichnis wird gewechselt.

### **ENV**

enthält den Namen einer Datei, die Kommandos zur Initialisierung für die Shellumgebung beim Bearbeiten von Shellscripts enthält. (z.B. die Datei `~/ .bashrc`)

### **FCEDIT**

benennt einen anderen Editor als `vi` als Standardeditor für das [fc-Shellkommando](#).

## **FIGNORE**

kann eine durch Doppelpunkte getrennte Liste von Dateiendungen enthalten, die bei der automatischen Dateinamenerweiterung ignoriert werden sollen. Beispielsweise kann diese Variable mit „.o:.bak:.old:~“ belegt werden.

## **GLOBIGNORE**

kann eine durch Doppelpunkte getrennte Liste von Namen und Mustern enthalten, die bei der Pfadnamenerweiterung ignoriert werden sollen.

## **histchars**

enthält zwei oder drei Zeichen zur Kontrolle der Wiederholung von Kommandos aus dem Kommandozeilenspeicher. Das erste Zeichen leitet eine Kommandozeilenerweiterung aus dem Kommandozeilenspeicher ein. Die Voreinstellung ist `!'. Das zweite Zeichen leitet die ``Schnellsubstituion" ein und ist mit ^ vorbelegt. Das optionale dritte Zeichen kennzeichnet einen Kommentar im Historykommando. Wenn es als erstes Zeichen eines Wortes auftaucht wird der Rest der Zeile bei der History-Erweiterung ignoriert. Das bedeutet nicht unbedingt, daß die Shell bei der weiteren Interpretation den Zeilenrest als Kommentar erkennt.

## **HISTCONTROL oder history\_control**

bestimmt, welche Kommandos in den Kommandozeilenspeicher geschrieben werden. Wenn die Variable das Wort `ignorespace' enthält, werden nur die Zeilen in den Speicher geschrieben, die nicht mit einem Leerzeichen beginnen. Wenn die Variable das Wort `ignoredups' enthält, werden alle Zeilen, die der zuletzt gespeicherten Zeile entsprechen, nicht gespeichert. `ignoreboth' ist die Kombination der beiden anderen Schalter. Wenn die Variable nicht gesetzt ist oder irgendeinen anderen Wert enthält, werden alle Kommandozeilen in den Kommandozeilenspeicher geschrieben.

## **HISTFILE**

benennt eine Datei, in die der Inhalt des Kommandozeilenspeichers beim Beenden der Shell automatisch gesichert wird. Der Kommandozeilenspeicher wird beim Start einer neuen Shell aus dieser Datei aufgefüllt. Voreinstellung für HISTFILE ist ~/.bash\_history Wenn Sie (z.B. im X Window System) mehrere Shells gleichzeitig mit derselben Sicherungsdatei benutzen, bleiben nur die Kommandozeilen der zuletzt beendeten Shell erhalten.

## **HISTFILESIZE**

bestimmt die Anzahl der im HISTFILE zwischen zwei Sitzungen gespeicherten Kommandozeilen. Alle älteren Zeilen gehen verloren.

## **HISTIGNORE**

kann eine durch Doppelpunkt getrennte Liste von Mustern enthalten, die auf Kommandozeilen angewendet werden, nachdem diese die Prüfung von HISTCONTROL bestanden haben. Es werden nur solche Kommandozeilen in die History übernommen, auf die das Muster nicht zutrifft. Zusätzlich zu den bei der Pfadnamenerweiterung üblichen Jokerzeichen bedeutet bei HISTIGNORE das Zeichen & die vorangehende Zeile in der History.

## **HISTSIZE**

setzt die Anzahl der im Kommandozeilenspeicher erinnerten Zeilen fest. Alle älteren Zeilen werden vergessen.

## **HOME**

ist das Heimatverzeichnis des aktuellen Benutzers. Dieses Verzeichnis wird in der Datei /etc/passwd bestimmt und von login automatisch in die Variable HOME geschrieben. Das

in HOME gespeicherte Verzeichnis ist der Standardwert für das [cd-Shellkommando](#).

## **HOSTFILE oder hostname\_completion\_file**

ist der Name einer Datei mit dem gleichen Format wie die Datei /etc/hosts. Die Einträge dieser Datei werden zur Hostnamenerweiterung verwendet.

## **IFS**

ist der „interne Feldseparator“. Alle Zeichen dieser Shellvariablen werden als Trenner von Wörtern erkannt. Die Standardbelegung für IFS ist Leerzeichen, Tabulator und Zeilenende.

## **IGNOREEOF**

hat nur eine Bedeutung, wenn die Shell interaktiv läuft. Wenn die Variable eine ganze Zahl enthält, so wird diese Anzahl von `EOF'-Zeichen (CTRL-D) für das Dateiende bzw. das Ende der Eingabe abgewartet, bevor die Shell verlassen wird. Zu jedem `EOF' wird der Hinweis ausgegeben, daß die Shell mit logout verlassen werden soll.

Wenn die Variable leer ist oder keine Zahl enthält, so ist die Voreinstellung 10. Wenn die Variable nicht gesetzt ist, führt jedes EOF-Zeichen sofort zum Verlassen der Shell.

## **INPUTRC**

enthält den Namen der Initialisierungsdatei für [readline](#).

## **LANG**

dient zur "weichen" Voreinstellung für alle Locale-Kategorien. Jede Einzelkategorie kann wirksam mit einem anderen Wert belegt werden.

## **LC\_ALL**

dient zur verbindlichen Einstellung aller Locale-Kategorien. Abweichende Einstellungen einzelner Kategorien werden von dem hier festgelegten Wert verdrängt.

## **LC\_COLLATE**

Diese Variable wird von der bash ausgewertet, um die Sortierfolge der Dateinamen bei der Pfadnamen-Expansion festzulegen.

## **LC\_MESSAGES**

Diese Variable wird von der bash ausgewertet, um das Locale für die Übersetzung der \$ Zeichenketten zu bestimmen.

## **MAIL**

enthält den Namen der Datei, in der die elektronische Post für den Benutzer gespeichert wird. Wenn diese Datei nicht leer ist, wird der Anwender darauf hingewiesen, daß Post für ihn da ist.

## **MAILCHECK**

spezifiziert die Zeitintervalle, nach denen die Shell prüft, ob Post da ist. Die Voreinstellung ist 60 Sekunden.

## **MAILPATH**

ist eine durch Doppelpunkt getrennte Liste von (absoluten) Dateinamen, in denen Post für den Benutzer ankommen kann. Zu jeder Datei kann eine durch `?' eingeleitete Zeichenkette angegeben werden, die auf dem Bildschirm ausgegeben wird, wenn in der entsprechenden Datei Post angekommen ist.

## **MAIL\_WARNING**

ist ein Schalter. Ist die Variable gesetzt (egal, welcher Wert), erfolgt eine Warnung, wenn auf die Postdatei zugegriffen wurde. Damit kann sowohl festgestellt werden, ob neue Mail

angekommen ist, als auch jeder (möglicherweise illegale) Lesezugriff auf die Mailbox überwacht werden.

Dieser Schalter funktioniert nur, wenn die Mailbox in einem Verzeichnis liegt, dessen Dateisystem die Zugriffszeit verwaltet (nicht Minix oder Extended1). In der Version 2.0 wird dieser Schalter nicht unterstützt.

## **OPTERR**

ist ein Schalter. Wenn die Variable den Wert 1 enthält, werden die Fehlermeldungen der `getopts`-Shellfunktion ausgegeben. Enthält sie eine 0, werden die Fehlermeldungen unterdrückt.

## **PATH**

enthält eine durch Doppelpunkt getrennte Liste von Verzeichnissen. Wenn ein einfaches Kommando nicht als internes Shellkommando erkannt wird und nicht mit komplettem Pfadnamen (das ist der Pfad vom aktuellen Verzeichnis oder vom Wurzelverzeichnis aus) angegeben wird, dann sucht die Shell in allen Verzeichnissen der `PATH`-Variablen nach einem Programm mit passendem Namen und führt es aus.

Ein einzelner Punkt anstelle des Wurzelverzeichnisses steht für das aktuelle Verzeichnis. Eine Tilde ``~'` steht für das [Heimatverzeichnis des Anwenders](#).

Der Standardpfad zu den ausführbaren Dateien eines Linux-Systems ist:

```
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/X11R6/bin:/usr/local/bin
```

Die Verzeichnisse `sbin` können für User ohne Systemverwaltungsaufgaben auch entfallen. Bei Installationen ohne X Window System kann das Verzeichnis `/usr/X11R6/bin` ebenfalls weggelassen werden. Durch die Installation zusätzlicher Software können Erweiterungen des Pfades um Verzeichnisse wie `/usr/opt/bin` oder `/home/postgres/bin` notwendig werden.

Die `PATH`-Variable wird in der Regel von der Systemverwalterin in der Datei `/etc/profile` für alle Login-Shells gemeinsam auf einen bestimmten Wert gesetzt. Dieser Wert kann vom Benutzer beliebig verändert werden. Die `PATH`-Variable kann nicht mit dem `unset`-Shellkommando gelöscht werden.

Aus Gründen der Systemsicherheit ist es sinnvoll, das aktuelle Verzeichnis (bezeichnet durch einen Punkt) nur als letztes der zu durchsuchenden Verzeichnisse anzugeben. Anderenfalls könnte ein Standardprogramm aus dem Systempfad versehentlich mit einem gleichnamigen Kommando im aktuellen Verzeichnis verwechselt werden.

## **PROMPT\_COMMAND**

benennt ein Kommando, das vor jeder Eingabeaufforderung automatisch ausgeführt wird.

## **PS1**

ist die (rohe) Zeichenkette, die als Eingabeaufforderung (Prompt) die Arbeitsbereitschaft der Shell anzeigt. Die Variable kann eine Reihe symbolischer Namen enthalten, die vor der Ausgabe nach den im Abschnitt [`Eingabeaufforderung'](#) erläuterten Regeln erweitert werden. Die Voreinstellung ist ``$'`.

## **PS2**

enthält die Zeichenkette für die sekundäre Eingabeaufforderung. Sie wird genau wie `PS1` erweitert. Die sekundäre Eingabeaufforderung erscheint, wenn zu einem gegebenen Kommando interaktiv über die Shell weitere Kommandos oder Parameter von der Tastatur gelesen werden sollen. (Zum Beispiel in einer `for`-Schleife) Die Voreinstellung ist ``>'`.

## PS3

enthält den Prompt, mit dem die Eingabe bei der `select`-Konstruktion angefordert wird ([siehe select](#)). Voreinstellung ist ``#?'`.

## PS4

wird zur Anzeige der erweiterten Kommandos mit der Option ``-x'` benutzt ([siehe set](#)). Voreinstellung ist ``+'`.

## TIMEFORMAT

In dieser Shellvariablen kann ein Formatstring abgelegt werden, der die Ausgabe der shellinternen Funktion [time](#) bestimmt. Ähnlich wie bei `printf` leitet das Prozentzeichen `%` die Platzhalter für Zeitinformation im String ein.

**%[p][l]R**

(real time) die tatsächlich während der Ausführung des Kommandos vergangene Zeit

**%[p][l]U**

(user time) die für die Ausführung des Kommandos im User-Modus verbrauchte Zeit

**%[p][l]S**

(system time) die für die Ausführung des Kommandos im Kernel-Modus verbrauchte Zeit

**%P**

die Auslastung der CPU durch das Kommando in Prozent

Das optionale Argument *p* gibt die Anzahl der Dezimalstellen an, mit denen die Sekundenangaben formatiert werden. Die maximale Genauigkeit und gleichzeitig die Voreinstellung beträgt drei Dezimalstellen.

Der optionale Buchstabe *l* in der Formatangabe veranlaßt die Ausgabe eines längeren Formates, in dem auch die Minuten enthalten sind.

Zeilenvorschübe im Formatstring können durch die Tastenfolge `^V ^J` erzeugt werden.

## TMOUT

Mit dieser Variablen kann die Shell veranlaßt werden, nach einer bestimmten Zeit ohne Benutzeraktivität, also ohne Eingabe, automatisch zu beenden. Nur wenn die Variable eine Zahl enthält, wird diese Anzahl Sekunden auf die Eingabe gewartet.

Die folgenden Variablen werden automatisch durch die Shell gesetzt. Die meisten können vom Benutzer nicht verändert direkt werden.

## BASH

beinhaltet den kompletten Pfadnamen der aktuellen Shell.

## BASH\_VERSION

ist eine Array-Variable mit den einzelnen Komponenten der bereits in `BASH_VERSION` zusammengefaßten Versionsnummer.

## BASH\_VERSION

beinhaltet die Versionsnummer der Shell.

## DIRSTACK

ist eine Array-Variable, die den aktuellen Verzeichnisstapel enthält. Die Reihenfolge der

Einträge entspricht der von `dirs` angezeigten. Es ist möglich, einzelne Einträge zu verändern. Um einen neuen Eintrag zu erzeugen oder einen existierenden zu entfernen sind die Shellkommandos `pushd` und `popd` zu verwenden. Sollte `DIRSTACK` durch `unset` gelöscht werden gehen die besonderen Eigenschaften verloren, auch wenn das Array anschließend wieder erzeugt wird.

## **EUID**

ist die effektive Benutzerkennung des Anwenders. Während der Ausführung von Programmen, bei denen das `SUID`-Bit gesetzt ist, wird die effektive Benutzerkennung des Eigentümers der Programmdatei gesetzt.

## **GROUPS**

ist eine Array-Variable mit den numerischen IDs sämtlicher Gruppen, denen der User angehört.

## **HISTCMD**

enthält die Nummer des aktuellen Kommandos so, wie sie in der History gespeichert wird.

## **HOSTTYPE**

enthält den Hostnamen des lokalen Rechners.

## **HOSTTYPE**

enthält eine Kennung zur Identifikation des Rechnertyps. Für Linux kommen nur die Typen ``i386'` und ``i486'` in Frage.

## **LINENO**

enthält die aktuelle Zeilennummer im Shellsript. Wenn die Variable innerhalb einer Scriptfunktion aufgerufen wird, entspricht die Zahl den bis zum Aufruf innerhalb der Funktion ausgeführten einfachen Kommandos. Außerhalb von Shellsripten ist diese Variable nicht sinnvoll belegt. Wenn die `LINENO`-Shellvariable mit dem `unset`-Kommando gelöscht wird, kann sie nicht wieder mit ihrer automatischen Funktion erzeugt werden.

## **MACHTYPE**

enthält die Typenbezeichnung des Betriebssystems für das die Shell erzeugt wurde. Für Intel-Linux ist das etwas wie `i586-pc-linux-gnu`

## **OLDPWD**

ist das zuletzt aktuelle Verzeichnis (wird ebenfalls vom `cd`-Shellkommando gesetzt).

## **OPTARG**

enthält das Argument zu der zuletzt von `getopts` ausgewerteten Option ([siehe getopts](#)).

## **OPTIND**

enthält den Index der zuletzt von `getopts` ausgewerteten Option ([siehe getopts](#)).

## **OSTYPE**

enthält den Namen des Betriebssystems, also in diesem Fall ``Linux'`.

## **PIPESTATUS**

ist eine Array-Variable mit den Status-Rückgabewerten aller Kommandos einer Pipeline.

## **PPID**

ist die Prozeßnummer des Elternprozesses der Shell.

## **PWD**

ist das aktuelle Verzeichnis, wie es vom `cd`-Shellkommando gesetzt wird.

## **RANDOM**

liefert bei jeder Abfrage einen neuen Pseudozufallswert. Die Folge von Pseudozufallszahlen kann durch eine Zuweisung an RANDOM initialisiert werden. Gleiche Initialwerte führen zu gleichen Zahlenfolgen.

## **REPLY**

wird vom Shellkommando `read` gesetzt, wenn keine andere Variable als Rückgabeparameter benannt ist ([siehe read](#)).

## **SECONDS**

liefert die Anzahl von Sekunden seit dem Start der aktuellen Shell. Wenn SECONDS ein Wert zugewiesen wird, erhöht sich dieser Wert jede Sekunde automatisch um eins.

## **SHELLOPTS**

enthält eine durch Doppelpunkt getrennte Liste aller aktiven Shelloptionen. Diese Variable kann bei der laufenden Shell nur gelesen werden. Die einzelnen Einträge lassen sich mit dem Shellkommando `set -o` verändern. Wird die Shellvariable in die Umgebung einer neu gestarteten Shell exportiert, übernimmt die neue Shell die Einträge vor dem Lesen anderer Initialisierungsdateien.

## **SHLVL**

steht für den Shell-Level. Bei jedem Aufruf einer neuen Shell in der Shell wird der Shell-Level um eins erhöht. Eine Möglichkeit, zwischen den Levels zu wechseln, besteht nicht.

## **UID**

ist die Benutzerkennung des aktuellen Anwenders. Diese Kennung ist in der Datei `/etc/passwd` dem Benutzernamen zugeordnet.

# **Eindimensionale Arrays**

Seit Version 2.0 unterstützt die `bash` eindimensionale, (nichtnegativ) numerisch indizierte Arrays. Die einzelnen Felder eines Arrays entstehen wie einfache Variable durch Zuweisung eines Wertes. Eine Deklaration und Typisierung ist möglich, aber nicht notwendig. Es gibt keine Beschränkung für die Länge eines Arrays.

Die Initialisierung eines Arrays kann durch die Zuweisung einer einzigen Liste erfolgen. Die Indizierung beginnt mit 0. Wenn bei der Initialisierung der Index nicht angegeben ist, wird automatisch der auf den höchsten bereits definierten Index folgende Wert belegt.

Bei der Verwendung von Array-Feldern muß der Bezeichner in geschweiften Klammern eingeschlossen werden, damit er von einfachen Variablen unterschieden werden kann. Die Symbole `@` und `*` indizieren das gesamte Array. Bei der ersten Form wird immer eine Liste aller Arrayeinträge geliefert. Die zweite Form liefert eine einzige Zeichenkette mit allen Einträgen, wenn sie in Anführungszeichen eingeschlossen wird.

Das folgende Beispiel zeigt die Verwendung einer Array-Variablen:

```
$ farben=(BLACK RED GREEN [4]=BLUE MAGENTA CYAN WHITE)
$ echo ${farben[5]}
MAGENTA
```



```

$ farben[3]=YELLOW
$ echo ${#farben[1]}
3
$ echo ${#farben[@]}
8
$ for i in ${farben[*]}; do echo $i; done
BLACK
RED
GREEN
YELLOW
BLUE
MAGENTA
CYAN
WHITE
$ for i in "${farben[*]}"; do echo $i; done
BLACK RED GREEN YELLOW BLUE MAGENTA CYAN WHITE
$

```

## Positionsparameter

Ein Positionsparameter wird durch eine positive ganze Zahl bezeichnet, also durch Ausdrücke wie \$1, \$2 ...referenziert. Die Positionsparameter werden beim Aufruf der Shell in der Kommandozeile oder durch das set-Shellkommando (mit den Optionen `-' oder `-', [siehe set](#)) gesetzt.

Wenn ein Positionsparameter mit mehr als einer Stelle (>9) bezeichnet werden soll, muß er in geschweifte Klammern gesetzt werden: z.B. \${12}.

## Spezialparameter

Einige Parameter haben eine besondere Bedeutung. Diese Parameter können nur gelesen werden. Eine Zuweisung an diese Parameter ist verboten.

\*

steht (als Parameter) für alle Positionsparameter von 1 an. In Anführungszeichen gesetzt, steht ``\$\*`` für ein einziges Wort, bestehend aus dem Inhalt aller Positionsparameter mit dem ersten IFS als Trennzeichen.

@

steht ebenfalls für alle Positionsparameter von 1 an. In Anführungszeichen gesetzt, wird es aber durch die Werte der einzelnen Positionsparameter (jeweils ein einzelnes Wort) ersetzt.

#

steht für die Anzahl der Positionsparameter.

?

liefert den Rückgabewert (Status) des zuletzt ausgeführten Kommandos.

-

steht für die Optionsflags (von set oder aus der Kommandozeile).



\$

steht für die Prozeßnummer der Shell.

!

steht für die Prozeßnummer des zuletzt im Hintergrund aufgerufenen Kommandos.

0

steht für den Namen des Shellskripts oder der Shell selbst, wenn kein Shellskript ausgeführt wird.

—

(Unterstrich) steht für das letzte Argument des zuletzt ausgeführten Kommandos (nach allen Erweiterungen).

## Erweiterung

Nachdem die Kommandozeile in einzelne Kommandos zerlegt ist, werden die Wörter jedes einzelnen Kommandos nach Token durchsucht, die zu ersetzen sind. Es gibt acht Formen der Erweiterung:

1.

Klammererweiterung { , }

2.

Tildenerweiterung ~

3.

Parameter- und Variablenerweiterung \${ }

4.

Kommandosubstitution ` ` oder \$( )

5.

Arithmetische Erweiterung \$( ( )) (vor Version 2.0 \$[ ])

6.

Prozeßsubstitution <( )

7.

Worttrennung

8.

Pfadnamenerweiterung \* ? [ ]

Nur durch Klammererweiterung, Worttrennung oder Pfadnamenerweiterung können neue Wörter in der Kommandozeile entstehen. Die Ergebnisse aller anderen Erweiterungen werden als einzelne Wörter interpretiert. Einzige Ausnahme ist der Spezialparameter `\$\_`, dessen Inhalt ungeachtet von Blanks als einziges Wort behandelt wird.

## Klammererweiterung

Die Klammererweiterung generiert aus einer in geschweiften Klammern eingeschlossenen Liste von Bausteinen mehrere Zeichenketten (Wörter). Wenn die Klammer von konstanten Zeichenketten eingeschlossen ist, werden diese Zeichenketten an jedes der erzeugten Wörter angefügt. Es können mehrere Klammern verschachtelt werden.

Die dem Beispiel zur `for`-Schleife zugrunde liegende Aufgabe kann durch Klammererweiterung auf einer einzigen Kommandozeile formuliert werden:

```
$ mkdir /usr/local/man/{man,cat}{1,2,3,4,5,6,7,8,9,n,l}
$ _
```

Dieses Beispiel zeigt, daß im Unterschied zur Pfadnamenerweiterung hier auch neue Dateinamen erzeugt werden können.

Die Klammererweiterung wird vor allen anderen Ersetzungsoperationen durchgeführt. Es handelt sich um eine reine Textoperation, es werden keine Zeichen aus den Bausteinen verändert. Das geschieht erst in den darauf folgenden Schritten.

Mit dieser Behandlung geschweifeter Klammern verhält sich die `bash` absichtlich anders als die Bourne-Shell, die solche Zeichen als Teil von Wörtern nicht verändert. Um die Kompatibilität zur `sh` wiederherzustellen, kann die Klammererweiterung durch eine Änderung der entsprechenden `set`-Option durch das Kommando `set +o braceexpand` abgeschaltet werden.

## Tildenerweiterung

Wenn ein Wort mit einer Tilde (`~`) beginnt, werden alle Buchstaben vor dem ersten Schrägstrich `/` (Slash) als Benutzername interpretiert. Diese werden dann durch den absoluten Pfad des Heimatverzeichnisses dieses Benutzers ersetzt. Wenn kein Benutzername angegeben ist, wird die Tilde durch den Inhalt der Umgebungsvariablen `HOME` bzw. das Heimatverzeichnis des aktuellen Anwenders ersetzt.

Wenn der Tilde ein `~+` folgt, wird das aktuelle Verzeichnis aus der Shellvariablen `PWD` eingesetzt. Folgt der Tilde ein `~-`, so wird mit dem letzten aktuellen Verzeichnis aus der Variablen `OLDPWD` erweitert.

Die Shell führt eine Tildenerweiterung bei der Zuweisung von Zeichenketten an Shellvariable durch. Dabei werden neben der Tilde am Wortanfang auch solche erkannt und ersetzt, die auf den Doppelpunkt `:` folgen. Auf diese Weise kann Tildenerweiterung auch für die Variablen `PATH`, `CDPATH` und `MAILPATH` durchgeführt werden.

Die Tildenerweiterung findet nach der Klammererweiterung und vor allen anderen Erweiterungen statt.

## Parametererweiterung

Das ``$'` Zeichen leitet Parametererweiterung, Kommandosubstitution und arithmetische Ersetzung ein. Der zu erweiternde Parameter kann in geschweiften Klammern eingeschlossen werden, um ihn von den folgenden Zeichen sicher abzugrenzen, die sonst als Teil des Variablennamens mißverstanden werden könnten.

**`${Parameter}`**

Der Wert des Parameters wird eingesetzt. Die geschweiften Klammern sind zwingend, wenn ein Positionsparameter mit mehr als einer Ziffer benannt ist oder wenn der Variablenname nicht eindeutig von den darauffolgenden Zeichen getrennt werden kann.

Die folgenden Konstruktionen stellen verschiedene Arten bedingter Parametererweiterung dar. Der mit *Wort* bezeichnete Teil ist seinerseits wieder Gegenstand von Tildenerweiterung, Parametererweiterung, Kommandosubstitution und arithmetischer Erweiterung. In allen Konstruktionen, die einen **Doppelpunkt** enthalten, wird der Parameter daraufhin getestet, ob er leer oder ungesetzt ist. Diese Konstruktionen können alle auch **ohne** den Doppelpunkt verwendet werden. In diesem Fall wird der Parameter nur daraufhin getestet, ob er ungesetzt bzw. gesetzt (auch leer!) ist.

Normalerweise wird eine Parametererweiterung an der Stelle im Shellscript durchgeführt, wo der entsprechende Wert das erste Mal benutzt wird. Bei der im folgenden vorgestellten Möglichkeit, Variable mit Defaultwerten zu belegen, ist es oft erwünscht, solche Belegungen gemeinsam an bestimmten Stellen des Scripts durchzuführen. Weil die Parametererweiterung nur als Bestandteil eines Kommandos oder einer Zuweisung durchgeführt wird, bietet sich zu diesem Zweck die `:`-Funktion an. Diese „Shellfunktion“ hat keine direkte Wirkung. Als Seiteneffekt können aber Parameter auf der Kommandozeile erweitert werden.

### **`${Parameter}:-Wort`**

Benutzung von Standardwerten. Wenn der *Parameter* ungesetzt oder leer ist, wird das *Wort* anstelle des gesamten Ausdrucks eingesetzt.

### **`${Parameter}:=Wort`**

Setzen von Standardwerten. Wenn der Parameter ungesetzt oder leer ist, wird der Inhalt des Wortes dem Parameter zugewiesen und der neue Parameter eingesetzt. Den Positionsparametern und den speziellen Parametern kann allerdings auch auf diese Weise kein Wert zugewiesen werden!

### **`${Parameter}:?Wort`**

gibt eine Fehlermeldung, wenn der Parameter leer oder ungesetzt ist. Der Inhalt von Wort wird als Fehlermeldung auf der Standardfehlerausgabe ausgegeben und eine nichtinteraktive Shell wird verlassen. Wenn der Parameter gültig gesetzt ist, wird sein Wert eingesetzt.

### **`${Parameter}:+Wort`**

erzwingt die Benutzung eines anderen Wertes. Wenn der Parameter weder leer, noch ungesetzt ist, wird der Inhalt von Wort eingesetzt. Sonst wird nichts eingesetzt.

### **`${Parameter}:Offset:Länge`**

wird durch den Abschnitt mit der angegebenen *Länge* aus der im *Parameter* enthaltenen Zeichenkette ersetzt, der an der Position *Offset* beginnt. Offset und Länge werden zuerst nach den Regeln der Arithmetischen Erweiterung aufgelöst. Ist Offset negativ, wird der Abschnitt vom Ende der Zeichenkette abgemessen. Die Länge darf nicht negativ werden.

Ist der Parameter `@`, so liefert der Ausdruck eine Liste mit der durch die Länge angegebenen Anzahl Positionsparameter, beginnend mit Parameter `$Offset`. Wenn der Parameter ein Array ist, wird eine Liste mit dem Inhalt der durch den Abschnitt definierten Felder geliefert.

### **`${#Parameter}`**

wird durch die Anzahl der Zeichen im Parameter ersetzt. Wenn als Parameter ``*'`` oder ``@`` angegeben werden, wird die gesamte Länge des erweiterten Parameters eingesetzt. Das ist die Längensumme aller Positionsparameter inklusive der Worttrenner.

### **`${Parameter#Wort}`**

und

### **`${Parameter##Wort}`**

Der Anfang der im *Parameter* enthaltenen Zeichenkette wird mit dem erweiterten *Wort* verglichen und der übereinstimmende Teil aus dem *Parameter* entfernt. Dabei werden Jokerzeichen (Wildcards) (``*'`, ``?'` und `[...]`) behandelt, wie bei der [Pfadnamenerweiterung](#) beschrieben. Im Fall der Konstruktion mit einem ``#'` wird das kürzeste passende Muster vom *Parameter* entfernt, im Fall der zwei ``##'` das längste.

Beispielsweise kann der Kommandoname ohne Pfadanteil aus einem absoluten Kommando durch die Konstruktion `basename=${pathname##*/}` aus dem Pfadnamen in `pathname` extrahiert werden; eine Aufgabe, für die normalerweise das Shellutility `basename` benutzt wird.

**`${Parameter%Wort}`**

und

**`${Parameter%%Wort}`**

Das Ende der im *Parameter* enthaltenen Zeichenkette wird mit dem erweiterten *Wort* verglichen und der übereinstimmende Teil aus dem *Parameter* entfernt. Die Jokerzeichen werden auch hier wie bei der Pfadnamenerweiterung behandelt. Bei der Konstruktion mit einem ``%'` wird das kürzeste passende Muster vom *Parameter* entfernt, bei der Konstruktion mit zwei ``%%'` das längste.

Beispielsweise kann analog zum vorhergehenden Beispiel der reine Pfadanteil eines absoluten Kommandos durch die Konstruktion `dirname=${pathname%/*}` aus der Zeichenkette in `pathname` gezogen werden.

**`${Parameter/Muster/String}`**

und

**`${Parameter//Muster/String}`**

Das größte auf das *Muster* passende Stück der Zeichenkette im *Parameter* wird durch den *String* ersetzt. In der einfachen Variante wird nur das erste passende Teilstück, in der doppelten jedes passende Teilstück ersetzt. Wenn das *Muster* mit `#` beginnt, muß es auf den Anfang der Zeichenkette passen, wenn es mit `%` beginnt, auf deren Ende. Wenn der *String* leer ist oder der letzte Teil vollständig fehlt, werden die auf das *Muster* passende Stücke entfernt.

Wenn die Positionsparameter oder ein Array als *Parameter* übergeben werden, wird die Ersetzung auf jeden einzelnen Positionsparameter beziehungsweise jedes Feld der Arrays angewendet und eine Liste mit den resultierenden Zeichenketten zurückgeliefert.

## Kommandosubstitution

Die Kommandosubstitution ermöglicht es, die Ausgabe eines Kommandos direkt einer Shellvariablen zuzuweisen. Es gibt zwei mögliche Formen der Kommandosubstitution:

**`$(Kommando)`** und

**``Kommando``**

Die Kommandosubstitution wird durchgeführt, indem die Standardausgabe des Kommandos anstelle des Parameters eingesetzt wird.

Wenn die Substitution in der „alten“ Form mit den Apostrophen (Backquote, nicht Hochkomma) durchgeführt wird, behält das Fluchtsymbol ``\`` nur dann seine Sonderfunktion, wenn es vor den Zeichen ``$'`, `` `'` oder ``\'` steht. Wenn das Kommando in der Klammerform aufgerufen wird, werden alle Zeichen unverändert belassen.

Kommandosubstitution kann auch verschachtelt werden. In der „alten“ Form muß vor den inneren Apostrophen ein Fluchtsymbol stehen.

Wenn eine Kommandosubstitution innerhalb von Anführungszeichen durchgeführt wird, wird die Ausgabe des Kommandos nicht mehr in einzelne Wörter geteilt.

## Arithmetische Erweiterung

Durch die arithmetische Erweiterung ist es möglich, mit Shellvariablen zu rechnen.

Durch die Konstruktion

**`$ ( (Ausdruck) )`**

wird anstelle des *Ausdrucks* das Ergebnis der darin formulierten arithmetischen Berechnungen eingesetzt. Außerdem wird in der Version 2.0 auch die ältere Form der Arithmetischen Erweiterung mit eckigen Klammern unterstützt:

**`$ [Ausdruck]`**

Die Berechnungen werden mit 'langen Ganzzahlwerten', wie sie in der Programmiersprache C verwendet werden, ausgeführt. Eine Überlaufkontrolle findet nicht statt. Die Division durch Null führt zu einem Fehler und kann mit der `trap`-Shellfunktion abgefangen werden.

Die folgenden Operatoren sind erlaubt (die Gruppierung entspricht der Prioritätshierarchie):

`+ -`

die Vorzeichen

`! ~`

die logische und die bitweise Negation

`* / %`

Multiplikation, Division und Modulo (Rest bei ganzzahliger Division)

`+ -`

Addition und Subtraktion

`<< >>`

bitweise Links- und Rechts-Shift-Operation

`<= >= < >`

die Vergleiche

`== !=`

gleich und ungleich

`&`

bitweise Addition

`^`

bitweise XOR (Exklusiv-ODER)

`|`

bitweise ODER

&&

logisch UND

||

logisch ODER

## **Zuweisungen**

durch die Zuweisungsoperatoren `=` `*` `/` `%` `+=` `-=` `<<=` `>>=` `&=` `^=` `|=` kann wie in der Programmiersprache C eine Zuweisung mit einer Rechenoperation verbunden werden; zum Beispiel der Ausdruck

```
let zahl=$((zahl+1))
```

kann durch den Zuweisungsoperator `+=` zu

```
let zahl+=1
```

verkürzt werden

Alle Shellvariablen sind als Operanden erlaubt. Die Berechnung macht natürlich nur Sinn, wenn es sich bei den Werten um gültige Zahlen handelt. Das Ganzzahlattribut für die Variable muß nicht gesetzt sein.

Die Operationen werden von links nach rechts (der Reihe nach) ausgeführt. Klammern werden erkannt und vorrangig behandelt.

Die kombinierten Zuweisungsoperationen sowie die logischen UND-, ODER- und Shift-Operationen gibt es erst seit `bash` Version 1.13.

Der Ausdruck wird behandelt, als ob er in Anführungszeichen stünde. Zusätzliche Anführungszeichen im Ausdruck werden ignoriert. Alle Wörter des Ausdrucks werden vor der Berechnung einer Parametererweiterung, Kommandosubstitution und Quotenreduktion unterzogen.

## **Prozeßsubstitution**

Eine dem Pipelining ähnliche Erweiterung wird durch die Prozeßsubstitution angeboten. Durch die beiden Konstruktionen

```
<(Subkommando)
```

```
>(Subkommando)
```

wird das *Subkommando* gestartet und sein Aus- bzw Eingabekanal mit einer automatisch erzeugten „named pipe“ oder FIFO-Datei verbunden. Auf der Kommandozeile wird die Konstruktion durch den Namen dieser FIFO-Datei ersetzt, so daß das eigentlich aufgerufene Kommando aus dieser Datei lesen bzw. dorthin schreiben kann.

Im Gegensatz zu einer normalen Pipeline können durch Prozeßsubstitution mehrere Pipelines gleichzeitig benutzt werden.

Die Subkommandos einer Prozeßsubstitution werden gleichzeitig mit den Parameter-, Kommando- und Arithmetikerweiterungen ausgeführt.

# Worttrennung


Die bisher beschriebenen Erweiterungen (Parametererweiterung, Kommandosubstitution und arithmetische Erweiterung) werden vor der weiteren Bearbeitung jeweils einer Worttrennung unterzogen, wenn die erweiterten Parameter nicht in Anführungszeichen stehen.

Bei der Worttrennung wird jedes der in der Shellvariablen `IFS` - dem internen Feldseparator - festgelegten Zeichen als Trenner behandelt. Wenn der Inhalt der `IFS`-Variablen exakt `SPACE TAB RETURN` (die Voreinstellung) ist, wird eine beliebige Folge dieser Zeichen als ein einziger Trenner behandelt. Anderenfalls führt jeder Separator zu einem neuen Wort. Wenn die `IFS`-Variable leer ist, wird keine Worttrennung durchgeführt.

Leere Token, wie sie zum Beispiel durch die Erweiterung leerer Parameter entstehen, werden entfernt, wenn sie nicht ausdrücklich als leere Zeichenkette in Anführungszeichen gesetzt sind.

Wenn keine Erweiterung durchgeführt wurde, findet auch keine Worttrennung statt.

## Pfadnamenerweiterung

Im Anschluß an die Worttrennung werden alle Wörter der Kommandozeile nach den Zeichen `*`, `?` und `[]` durchsucht. Jedes Wort, das ein solches Zeichen enthält, wird als Muster für eine Pfadnamenerweiterung benutzt. Es werden alle Dateien und Verzeichnisse des aktuellen Verzeichnisses mit dem Muster verglichen und anstelle des Musters eine alphabetisch  geordnete Liste aller passenden Pfadnamen eingesetzt.

Wenn kein passender Pfadname gefunden wurde und die Shelloption [`nullglob`](#) nicht aktiv ist, bleibt das Wort unverändert. Andernfalls wird es aus der Kommandozeile entfernt.

Bei der Pfadnamenerweiterung werden Datei- oder Verzeichnisnamen, die mit einem Punkt beginnen nur dann berücksichtigt, wenn die Shelloption `dotglob` aktiv ist. Der die Verzeichnisnamen trennende Schrägstrich ``/'` (Slash) kann durch kein Jokerzeichen ersetzt, sondern muß immer ausdrücklich angegeben werden.

Die Liste der durch Pfadnamenerweiterung erzeugten Token kann eingeschränkt werden, indem in der Umgebungsvariablen `GLOBIGNORE` eine Liste mit entsprechenden Mustern angegeben wird. Die Muster werden wie die zur Pfadnamenerweiterung selbst auf die bei der ursprünglichen Erweiterung erzeugten Token angewandt. Es werden alle Token aus der Liste entfernt, auf die ein Muster aus `GLOBIGNORE` paßt.

Ist `GLOBIGNORE` gesetzt wird automatisch auch die Shelloption `dotglob` aktiviert. Wird `GLOBIGNORE` durch `unset` wieder aus der Umgebung entfernt, dann wird auch `dotglob` wieder abgeschaltet.

Die Jokerzeichen (Wildcards) zur Pfadnamenerweiterung haben die folgende Bedeutung:

**\***

paßt auf alle Zeichenketten, einschließlich der leeren Zeichenkette

**?**

paßt auf jedes einzelne Zeichen, außer auf das Zeilenende

[...]

paßt auf alle der in den eckigen Klammern eingeschlossenen Zeichen. Ein Paar Zeichen mit einem Minuszeichen dazwischen ist ein Bereich. Dieser Bereich enthält alle Zeichen, die nach lexikalischer Ordnung zwischen den beiden begrenzenden Zeichen angeordnet sind. Wenn das erste Zeichen ein Ausrufezeichen (!) oder ein Caret (^) ist, paßt das Muster auf alle Zeichen, die nicht eingeschlossen sind. Das Minuszeichen oder die eckige Klammer selbst kann in den Bereich eingeschlossen werden, indem es zusätzlich als einzelnes Zeichen vor oder nach dem Bereich angegeben wird.

Wenn die `set`-Option `braceexpand` gesetzt ist, werden in geschweiften Klammern eingeschlossene, durch Komma getrennte Listen von Wortteilen zu einer Liste von Wörtern erweitert wie in der C-Shell.

Zum Beispiel wird ``*.{c,h}'` zu ``*.c *.h'` erweitert.

## Quotenreduktion

Nach allen Erweiterungen werden die unquotierten Fluchtsymbole und Apostrophe entfernt.

## Synonyme

Die `bash` unterhält eine Liste von Synonymen, die mit den Shellkommandos `alias` und `unalias` verwaltet werden kann. Das erste Wort jeder Kommandozeile wird daraufhin untersucht, ob es mit einem Synonym übereinstimmt. Wenn das der Fall ist, wird das Wort durch das Synonym ersetzt. Der Synonymname muß ein gültiger Name sein, der Ersetzungstext muß den Regeln der Shellgrammatik entsprechen.

Das erste Wort des Ersetzungstextes wird wieder auf ein Synonym untersucht. Es wird aber dasselbe Synonym nicht rekursiv ersetzt, so daß ein bereits existierendes Kommando über ein Synonym unter dem normalen Namen mit speziellen Optionen aufgerufen werden kann. Beispielsweise kann mit ``alias ls='ls -F' '` bei jedem Aufruf von `ls` die Option `-F` automatisch gesetzt werden.

Wenn das letzte Zeichen des Ersetzungstextes ein Blank ist, wird das dem Synonym folgende Wort wieder auf ein Synonym untersucht.

Es gibt keine Möglichkeit, in dem Ersetzungstext Argumente zu verwenden. Wenn Argumente gebraucht werden, sollte anstelle eines Synonyms eine Scriptfunktion benutzt werden.

Synonyme werden mit dem `alias`-Shellkommando erzeugt und aufgelistet. Mit dem `unalias`-Shellkommando werden sie wieder gelöscht ([siehe alias](#) und [unalias](#)).

Synonyme sollten nicht innerhalb von Scriptfunktionen definiert werden. Außerhalb der Scriptfunktion definierte Synonyme können aber durchaus in der Scriptfunktion verwendet werden.

## Signale

Wenn die `bash` interaktiv arbeitet, ignoriert sie `SIGTERM` und `SIGQUIT`. `SIGINT` wird für das `trap`-Shellkommando abgefangen, aber nicht von der Shell selbst bearbeitet ([siehe trap](#)).



Die Signale zur Jobkontrolle SIGTTIN, SIGTTOU und SIGTSTP werden von der Shell selbst ebenfalls ignoriert.

Mit dem suspend-Shellkommando kann die Shell bis zu einem SIGCONT-Signal angehalten werden.

Hintergrundjobs ignorieren die Signale SIGINT und SIGQUIT.

Jobs, die durch Kommandosubstitution von der Shell aufgerufen wurden, reagieren nicht auf die Jobkontrollsignale SIGTTIN, SIGTTOU und SIGTSTP.

## Eingabeaufforderung

Wenn die Shell interaktiv arbeitet und auf die Eingabe einer neuen Kommandozeile wartet, signalisiert sie ihre Bereitschaft durch die Ausgabe eines Zeichens oder einer kurzen Meldung. Diese Meldung wird auch als **Prompt** bezeichnet. Sie kann durch den Anwender selbst gestaltet werden. Dazu muß die gewünschte Meldung in die Shellvariable PS1 oder PS2 geschrieben werden.

Die Shell benutzt zwei Prompts für Eingabeaufforderungen. Der erste erscheint bei jeder interaktiven Eingabeaufforderung. Der zweite erscheint, wenn die Shell zur Vervollständigung eines vorangegangenen Kommandos noch weitere Eingaben vom Benutzer erwartet.

Für die Gestaltung des Prompts stehen die folgenden Sonderzeichen zur Verfügung:

<b>\a</b>	ein Alarmton
<b>\d</b>	das aktuelle Datum
<b>\e</b>	das Escape-Zeichen
<b>\h</b>	der Hostname (Netzwerkname des Rechners)
<b>\n</b>	ein Zeilenende
<b>\s</b>	der Name der aktuellen Shell (Inhalt vom Parameter 0)
<b>\t</b>	die aktuelle Zeit im 24-Stunden-Format
<b>\T</b>	die aktuelle Zeit im 12-Stunden-Format
<b>\u</b>	der Benutzername
<b>\v</b>	die Versionsnummer der Shell

**\v**

die Versionsnummer mit Patchlevel

**\w**

das aktuelle Verzeichnis

**\W**

auch das aktuelle Verzeichnis, ohne Pfad

**\@**

die aktuelle Zeit im 12-Stunden-Format ohne Sekunden mit am/pm Ergänzung

**\#**

die (absolute) Nummer des aktuellen Kommandos

**\!**

die Nummer, unter der das aktuelle Kommando im Kommandozeilenspeicher geführt wird

**\\$**

der „Standardprompt“. Ein \$-Zeichen für normalsterbliche Anwender, ein # für die Superuserin (ruth)

**\nnn**

das Zeichen mit dem (oktalen) Code *nnn*

**\**

der Backslash `\'

**\[**

der Anfang einer Sequenz nichtdruckbarer Zeichen

**\]**

das Ende einer Sequenz nichtdruckbarer Zeichen

## Wenn alles getan ist

Wenn die Shell alle Ersetzungen in der Kommandozeile vorgenommen und alle Umleitungen vorbereitet hat, ist der Zeitpunkt gekommen, auf den der Anwender die ganze Zeit gewartet hat. Die Shell versucht, das Kommando auszuführen. Dazu muß sie es aber erst lokalisieren. Das erste passende Kommando wird ausgeführt.


Als Kommandoname wird immer das erste Wort eines einfachen Kommandos erkannt. Ein Kommandoname kann mit Pfadnamen in einem Verzeichnis (absolut oder relativ) angegeben werden. Die Shell erkennt das an (mindestens) einem Slash `/` im Kommandonamen. Wenn kein Verzeichnis angegeben ist, versucht die Shell selbst, das Kommando zu finden. Dazu wird der Kommandoname zuerst in der [Hashtabelle](#) gesucht, dann wird er mit den Synonymen, mit den Scriptfunktionen und schließlich mit den Shellfunktionen verglichen.

Wenn auf diese Weise kein Programm dieses Namens gefunden wird, werden alle in der PATH-Umgebungsvariablen aufgeführten Verzeichnisse nach einer ausführbaren Datei dieses Namens durchsucht. Wenn auch hier kein passendes Kommando gefunden wird, gibt die Shell eine Fehlermeldung aus.

Wenn die Shell das Kommando lokalisieren kann, wird ein Argumentvektor zusammengestellt. Die

Positionsvariable 0 wird mit dem vollständigen Pfadnamen des Kommandos belegt, die übrigen Argumente (wenn welche vorhanden sind) werden in den folgenden Positionsparametern gespeichert und in der Laufzeitumgebung des Kommandos, dem Environment, gespeichert. Dieser Argumentvektor kann dann beispielsweise in einem C-Programm durch die Funktion `main(int argc, char* argv)` übernommen und später ausgewertet werden.

Wenn die Ausführung fehlschlägt, weil die Datei keine ausführbaren Binärdaten enthält, versucht die `bash` automatisch, diese Datei als Shellsript auszuführen. Dazu wird eine neue Shell (Subshell) erzeugt, die wie bei einem Aufruf in der Kommandozeile neu (durch die in der ENV-Umgebungsvariablen angegebene Datei) initialisiert wird.

Häufig beginnen Scriptprogramme mit den Zeichen ``#!'`, gefolgt von einem Programmnamen. An dieser Konstruktion erkennt der Kernel selbst, daß es sich nicht um ein Binärprogramm handelt.  Beispielsweise leitet die Zeile `#!/bin/bash -e` ein `bash`-Shellsript ein, das sofort abbricht, wenn eines der aufgerufenen Kommandos einen Status `!= Null` liefert.

## Eingebaute Shellkommandos

:

Syntax: **:** [*Argument...*]

Das Shellkommando **:** hat keinen direkten Effekt. Im Gegensatz zum Kommentar werden aber die „Argumente“ wie bei normalen Kommandos erweitert. Selbstverständlich muß das **:**-Shellkommando an einer für Kommandos zulässigen Position in der Kommandozeile stehen.

Seine Bedeutung hat dieses Kommando in Shellscripts, wo bestimmte Parametererweiterungen, beispielsweise Belegung von Shellvariablen mit [Default-Werten](#), unabhängig von einem einzelnen Kommando ausgeführt werden sollen.

## alias

Syntax: **alias** [-p] [*Name[=Kommando]...*]

Mit dem Shellkommando `alias` können einfache Kommandos mit benutzerdefinierten Namen (Synonymen) belegt werden. Wenn ein Name und ein einfaches Kommando angegeben sind, wird der Name als Synonym für das Kommando gespeichert. Besteht das Kommando aus mehreren Wörtern, muß es in Hochkommata oder Anführungszeichen eingeschlossen werden.

Ohne Argument oder durch Angabe der Option `-p` werden alle definierten Synonyme aufgelistet. Wenn nur ein Name angegeben ist, wird das Synonym für diesen Namen angezeigt. Der Rückgabewert von `alias` ist Null (wahr), wenn der Name als Synonym für ein Kommando steht.

Ein mit `alias` definiertes Synonym kann mit [unalias](#) wieder gelöscht werden.

# bg

Syntax: **bg** [*Jobspezifikation*]

Das Kommando **bg** startet einen (mit ^Z) angehaltenen Prozeß (Job) im Hintergrund. Wenn keine [Jobspezifikation](#) angegeben wurde, wird der zuletzt angehaltene Job gestartet.

Ein im Hintergrund laufender Job wird automatisch angehalten, wenn er vom Terminal lesen will.

# bind

Syntax: **bind** [-m *Keymap*] [-lpsSvV] [-q *Kommandobezeichnung*] [-r *Tastenfolge*]

**bind** [-m *Keymap*] [-f *Dateiname*]

**bind** [-m *Keymap*] [*Tastenfolge: Kommandobezeichnung*]

Mit dem **bind**-Shellkommando kann die Tastenbelegung der **readline**-Editorfunktionen angezeigt oder neu definiert werden.

Die **readline**-Funktionen unterstützen mehrere Editoremulationen. Dazu werden mehrere Keymaps verwaltet.

Die Syntax einer Tastaturbelegung entspricht der für die Datei [/.inputrc](#).

Die Optionen haben die folgende Bedeutung:

-m **Keymap**

wählt die bezeichnete Befehlstabelle *Keymap* zur Anzeige bzw. zur Veränderung; die folgenden Keymaps werden von **readline** verwaltet:

**emacs**

**emacs-standard**

**emacs-meta**

**emacs-ctlx**

**vi**

**vi-move**

**vi-command**

**vi-insert**

-f **Datei**

liest die Tastaturbelegung aus der *Datei*

-l

gibt eine Liste aller möglichen Kommandobezeichnungen (**readline**-Funktionen) aus

-p

gibt eine Liste aller belegten Tastenkombinationen und ihrer Funktionen im Format zum Wiedereinlesen auf die Standardausgabe

-P

gibt eine informative Liste aller Kommandobezeichnungen und ihrer Belegungen aus

-q **Kommando**

gibt die Tastenkombination aus, die zur Ausführung des angegebenen *Kommando* führt

-v

gibt eine Liste der Schalter und Variablen von `readline` im Format zum Wiedereinlesen auf die Standardausgabe

-V

gibt eine informative Liste aller aktuellen Schalter- und Variableneinstellungen von `readline` aus

## break

Syntax: **break** [*n*]

Das `break`-Shellkommando bricht eine `for`-, `while`- oder `until`-Schleife ab. Wenn eine Zahl *n* als Argument angegeben ist, werden *n* verschachtelte Schleifen abgebrochen. Ist die Anzahl größer als die Zahl der umgebenden Schleifen, werden alle umgebenden Schleifen verlassen.

## builtin

Syntax: **builtin** *Shellkommando* [*Argument...*]

Das Shellkommando `builtin` führt ein anderes eingebautes Shellkommando aus, auch wenn es durch ein Synonym oder eine gleichnamige Scriptfunktion verdeckt ist. Eine mit ``enable -n'` abgeschaltete Shellfunktion kann auch mit dem `builtin`-Kommando nicht aufgerufen werden.

## bye

Wird seit Version 1.14 nicht mehr unterstützt. Siehe [exit](#).

## cd

Syntax: **cd** [-LP] [*Verzeichnis*]

Das Shellkommando `cd` dient zum Wechseln des aktuellen *Verzeichnisses*. Wenn kein Verzeichnis angegeben ist, wird in das HOME-Verzeichnis gewechselt. In der Shellvariablen `CDPATH` kann eine Liste von (durch Doppelpunkt getrennten) Verzeichnissen angegeben werden, in denen das Verzeichnis gesucht wird. Ein Verzeichnisname, der mit einem ``/'` beginnt, wird als absoluter Name behandelt und nur vom Wurzelverzeichnis aus gesucht.

Über symbolische Links können unterschiedliche Wege in das gleiche Zielverzeichnis führen. Die Shell berücksichtigt diese Möglichkeit und unterscheidet intern zwischen dem ``physischen" und dem ``logischen" Arbeitsverzeichnis. Normalerweise wird bei relativen Pfadangaben in der Shell immer vom logischen Arbeitsverzeichnis ausgegangen. Die Option `-P` führt dazu, daß beim Wechseln durch

Pfade, in denen symbolischen Links enthalten sind, das aktuelle Arbeitsverzeichnis in das ``physische" Zielverzeichnis wechselt. Dieses Verhalten kann auch generell durch das Shellattribut `physical` mit dem Kommando `set` voreingestellt werden.

Die Option `-L` sollte das Folgen symbolischer Links in das logische Zielverzeichnis erzwingen, auch wenn das Shellattribut `physical` gesetzt ist. In der Version 2.01 funktioniert dieser Schalter jedoch nicht.

Für relative Pfadangaben ist das aktuelle Verzeichnis selbst unter der Bezeichnung ``.`` ansprechbar; das im Verzeichnisbaum nächsttiefere Verzeichnis heißt `..`` und das letzte aktuelle Verzeichnis (`OLDPWD`) heißt `-``.

## command

Syntax: **command** [-pVv] *Kommando* [*Argument...*]

Das Shellkommando `command` führt das angegebene (einfache) Kommando ohne die normale shellinterne Identifizierung aus. Dadurch werden nur die fest eingebauten Shellfunktionen und die Dateien aus den Verzeichnissen in `PATH` ausgeführt.

`-p`

schränkt die Suche nach dem Kommando auf den Standardpfad ein; auf diese Weise kommen nur die Standardsystemkommandos zur Ausführung

`-v`

(verbose) wortreich (Siehe Langenscheidts Taschenwörterbuch „Englisch“)

`-V`

(even more verbose) wortreicher ('tschuldigung')

## continue

Syntax: **continue** [*n*]

Mit der Shellfunktion `continue` wird der aktuelle Schleifendurchlauf einer `for`-, `while`- oder `until`- Schleife sofort unterbrochen und mit dem nächsten Schleifendurchlauf angefangen. Wenn als Argument eine Zahl (größer als Null) angegeben ist, wird diese Anzahl umgebender Schleifen abgebrochen. Wenn die Zahl größer als die Zahl der umgebenden Schleifen ist, werden alle umgebenden Schleifen unterbrochen und mit dem nächsten Durchlauf der äußersten Schleife fortgefahren.

## declare

Syntax: **declare** [-afFrx] [-p] [*Name*[=*Wert*]]

Das Shellkommando `declare` erzeugt eine Shellvariable und/oder setzt die Attribute der Variablen. Wenn kein Name angegeben ist, werden die Werte aller Variablen angezeigt.

Das Kommando erlaubt folgende Optionen:

-a

deklariert eine Array-Variable

-f

gibt die vollständige Definition aller oder der angegebenen Funktionen aus

-F

gibt eine Liste von `declare`-Kommandos aus, mit denen alle definierten Funktionen angezeigt werden können

-i

setzt den Typ der Variablen auf Ganzzahl; wenn dieser Variablen ein Wert zugewiesen wird, findet eine arithmetische Auswertung des zugewiesenen Ausdrucks statt

-p

gibt die Belegung aller oder der angegebenen Variablen aus, alle anderen Optionen werden ignoriert

-r

setzt die Variable(n) auf 'nur Lesen' Status

-x

markiert die Variable für automatischen Export in alle Subshellumgebungen

Wenn die Optionen mit '+' anstelle von '-' gesetzt werden, wird das entsprechende Merkmal abgeschaltet.

Wenn die Shellfunktion innerhalb einer Funktionsdefinition benutzt wird, ist die Variable lokal zu dieser Funktion, genauso als ob sie mit der Shellfunktion `local` definiert wäre.

Die **typeset**-Shellfunktion ist identisch mit der `declare`-Shellfunktion.

## dirs

Syntax: **dirs** [-c|pv] [+n] [-n]

Die Shellfunktion `dirs` gibt eine Liste der im Verzeichnisstapel gespeicherten Verzeichnisse aus. Die Bearbeitung dieses Verzeichnisstapels findet mit den Shellkommandos `pushd` und `popd` statt ([siehe popd](#)).

+n

zeigt das *n*-te Verzeichnis vom Boden des Verzeichnisstapels an

-n

zeigt das *n*-te von der Spitze des Verzeichnisstapels an

-c

löscht den gesamten Verzeichnisstapel

-l

veranlaßt die Ausgabe des vollständigen Pfadnamens der sonst durch Tilde abgekürzten Unterverzeichnisse des Heimatverzeichnisses

-p

gibt den Inhalt des Verzeichnisstapels zeilenweise aus

-v

gibt den Inhalt des Verzeichnisstapels zeilenweise mit Indexzahlen aus

## disown

Syntax: **disown** [-h] [*Jobspezifikation*]

Durch `disown` wird der durch die *Jobspezifikation* angegebene, im Hintergrund laufende Prozeß aus der Tabelle aktiver Jobs entfernt. Das führt dazu, daß der Prozeß beim Verlassen der Shell nicht mehr beachtet wird und kein SIGHUP an ihn weitergeleitet wird. Durch Angabe der Option `-h` wird lediglich die Weitergabe von Signalen unterdrückt, der Prozeßeintrag in der Jobtabelle bleibt aber bestehen. Wenn keine *Jobspezifikation* angegeben ist, wird der aktuelle Job [aktuelle Job](#) verarbeitet.

## echo

Syntax: **echo** [-neE] [*Argument...*]

Das `echo`-Shellkommando gibt die Argumente (durch Leerzeichen getrennt) auf die Standardausgabe. Es gibt auch ein externes `echo`-Kommando, das mit dem eingebauten Shellkommando der `bash` identisch ist.

Wenn die Option `-n` gesetzt ist, wird die Ausgabe nicht durch ein Zeilenendezeichen abgeschlossen.

Wenn die Option `-e` gesetzt ist, werden die folgenden Sonderzeichen zur Formatierung der Ausgabe erkannt:

<code>\a</code>	Alarm (Piep)
<code>\b</code>	Schritt zurück
<code>\c</code>	kein Zeilenende
<code>\f</code>	Seitenvorschub
<code>\n</code>	Zeilenende
<code>\r</code>	Wagenrücklauf
<code>\t</code>	(horizontaler) Tabulator
<code>\v</code>	vertikaler Tabulator
<code>\\</code>	das Zeichen <code>`\`</code>
<code>\nnn</code>	das Zeichen mit dem (oktalen) Code <i>nnn</i>



# enable

Syntax: **enable** [-adnps] [-f *Objektdatei*] [*Kommando*...]

Das Shellkommando **enable** ermöglicht es, Shellfunktionen ab- und wieder anzuschalten. Auf diese Weise kann anstelle eines (internen) Shellkommandos das gleichnamige (externe) Kommando aus einem Binärverzeichnis ausgeführt werden.

Wenn die Option ``-n'` gesetzt ist, wird das Shellkommando abgeschaltet. Sonst wird das Shellkommando eingeschaltet.

Wenn kein Argument oder die Option ``-p'` angegeben ist, wird eine Liste der aktiven Shellkommandos ausgegeben. Mit der Option ``-a'` werden auch die abgeschalteten Shellkommandos mit aufgelistet. Die Option ``-s'` unterdrückt die Auflistung der nicht von POSIX vorgesehenen Shellkommandos.

Seit Version 2.0 der **bash** können zur Laufzeit weitere Shellfunktionen dynamisch hinzugelinkt werden. Dazu wird mit der Option ``-f'` der Name der *Objektdatei* und zusätzlich der Name für das *Kommando* angegeben. Mit der Option ``-d'` wird ein geladenes Shellkommando wieder entfernt.

# eval

Syntax: **eval** [*Argument*...]

Das **eval**-Shellkommando fügt die Argumente zu einer Kommandozeile zusammen, die ausgeführt wird, ohne die Shell zu verdrängen. Sinn dieses Shellkommandos ist es, eine Kommandozeile mehrfach der Parametererweiterung zu unterziehen.

Wenn in einem Shellsript Variablen eingesetzt werden müssen, die ihrerseits wieder Variable enthalten, oder wenn aufgrund der Reihenfolge ihrer Ausführung die gewünschte Erweiterung nicht erzielt wird, ist dieses Shellkommando das Mittel der Wahl.

# exec

Syntax: **exec** [-cl] [-a *Name*] [*Kommando*] [*Argument*...]

Normalerweise startet die Shell ein Programm mit dem *fork*-Systemaufruf und wartet im Hintergrund, bis das Programm beendet wird. Danach übernimmt die Shell wieder die Kontrolle über das Terminal.

Das Shellkommando **exec** führt ein Kommando mit den angegebenen Argumenten aus, ohne einen Kindprozeß zu erzeugen. Das heißt, die aufrufende Shell wird verdrängt und damit beendet. Auch wenn das Kommando aus irgendwelchen Gründen nicht ausgeführt werden kann, wird die Shell beendet, wenn die Shelloption **execfail** (→ [shopt](#)) nicht gesetzt ist.

Die Argumente werden als Optionen und Positionsparameter an das Kommando weitergegeben. Wenn die Option ``-l'` angegeben ist, wird ein `-` als nulltes (!) Argument der Kommandozeile an das

Kommando weitergegeben. Das ist die Art, wie `login` ein Programm aufruft. Mit der Option ``-l'` kann ein beliebiger andere Name als nulltes Argument der Kommandozeile in die Umgebung des *Kommandos* eingetragen werden.

Die Option ``-c'` veranlaßt die Ausführung des *Kommandos* in einer leeren Umgebung.

Wenn kein Kommandoname angegeben ist, werden die Ein-/Ausgabe-Umleitungen, die mit dem `exec`-Shellkommando gegeben werden, auf die aufrufende Shell angewendet.

## exit

Syntax: **exit** [*n*]

Das Shellkommando `exit` verläßt die Shell mit dem Status *n*. Wenn kein Status angegeben ist, wird der Status des zuletzt ausgeführten Kommandos (in der Shellvariablen ``?'`) zurückgegeben.

Die `exit`-Shellfunktion erzeugt ein `EXIT`-Signal (0), das mit dem `trap`-Shellkommando abgefangen und als letzte Aktion der Shell behandelt werden kann.

Bis zur Version 1.13 konnte die Shellfunktion `exit` auch unter dem Namen **bye** aufgerufen werden.

## export

Syntax: **export** [`-nfp`] [*Name*[=*Wert*]]

Bei der Ausführung von Programmen durch die Shell werden in der Regel neue Prozesse erzeugt. Diese Prozesse "erben" von der Shell eine Umgebung (Environment), in der verschiedene „globale“ Variablen und Funktionen enthalten sein können ([siehe Shellvariable](#)). Diese können vom Prozeß ausgewertet werden.

Es werden aber nicht alle Shellvariablen, sondern nur die besonders für den Export bestimmten Umgebungsvariablen und Funktionen aus der Shellumgebung in die Umgebung eines neuen Prozesses kopiert.

Das `export`-Shellkommando schreibt bereits existierende Shellvariable in die Prozeßumgebung und macht sie so zu Umgebungsvariablen. Wenn die Option ``-n'` gesetzt ist, wird die Variable aus der Prozeßumgebung entfernt, innerhalb der Shell bleibt sie als Shellvariable erhalten. Um Funktionen zu exportieren, muß die Option ``-f'` benutzt werden.

Wenn keine Namen angegeben sind oder die Option ``-p'` gesetzt ist, werden alle für den Export bestimmten Shellvariablen mit ihren Werten angezeigt. Eine komplette Liste aller Umgebungsvariablen können Sie sich auch mit dem [printenv-Kommando](#) anzeigen lassen. In dieser Liste werden auch die Variablen angezeigt, die die Shell beim Start mit ihrer eigenen Umgebung erhalten hat.

Wenn zu einem Variablennamen beim Aufruf von `export` eine Zuweisung erfolgt, wird eine existierende Variable mit diesem Wert belegt oder eine neue mit diesem Wert erzeugt.

Die mit Magnetbandgeräten arbeitenden Kommandos versuchen beispielsweise die Gerätedatei des Streamers aus der Umgebungsvariablen `TAPE` zu lesen. Wenn Sie beispielsweise einen (den ersten)

SCSI-Streamer als Standardgerät für alle Bandoperationen bestimmen wollen, können Sie mit dem folgenden Kommando die Umgebungsvariable erzeugen:

```
$ export TAPE=/dev/st0
$ _
```

## fc

Syntax: **fc** [-e *Editor*] [-n|*r*] [*Anfang*] [*Ende*]  
**fc** -s [*Muster=Ersatz*] [*Kommandozeile*]

Mit dem Shellkommando **fc** (fix command) können einzelne Kommandozeilen aus dem Kommandozeilenspeicher, aber auch ganze Bereiche des Kommandozeilenspeichers editiert und danach ausgeführt werden. Als Editor wird der mit der Option `-e` spezifizierte Editor benutzt oder der in der Shellvariablen `FCEDIT` bestimmte oder schließlich der Standardeditor `vi`, wenn kein anderer Editor bestimmt wird.

In der ersten Form werden die Kommandozeilen von *Anfang* bis *Ende* in den Editor geladen. Anfang und Ende können als Zeichenkette (in Übereinstimmung mit dem Anfang der gewünschten Kommandozeile) oder als Zahl (die absolute Position des Kommandos im Kommandozeilenspeicher) angegeben werden. Eine negative Zahl bestimmt Anfang und Ende relativ zum aktuellen Kommando.

Wenn kein Ende gesetzt ist, wird nur das Kommando am Anfang editiert. Wenn auch kein Anfang gesetzt ist, wird das letzte Kommando genommen.

Wenn die Option `-l` gesetzt ist, wird der entsprechende Bereich von Kommandozeilen angezeigt, anstatt ihn zu editieren. Wenn zusätzlich noch die Option `-n` gesetzt ist, wird die Ausgabe der Zeilennummern vor den Kommandozeilen unterdrückt.

Wenn die Option `-r` gesetzt ist, werden die Kommandozeilen in umgekehrter Reihenfolge in den Editor geladen.

Wenn das Shellkommando **fc** in der zweiten Form aufgerufen wird, ersetzt es das *Muster* in der *Kommandozeile* durch *Ersatz*. Wenn kein Muster/Ersatz-Paar angegeben wird, kommt die Kommandozeile unverändert zur Ausführung.

## fg

Syntax: **fg** [*Jobspezifikation*]

Das Shellkommando **fg** bringt einen (mit `^Z`) angehaltenen Prozeß im Vordergrund zum Laufen. Wenn keine [Jobspezifikation](#) angegeben ist, wird der zuletzt angehaltene Job im Vordergrund gestartet.

## getopts

Syntax: **getopts** *Optionen Variable* [*Argumente*]

Die `getopts`-Shellfunktion kann in ShellscripTEN verwendet werden, um die Kommandozeile nach (konventionell) gültigen Optionen und Argumenten zu durchsuchen.

Wenn ein ShellscripT als Kommando aufgerufen wird, kann es auf der Kommandozeile Optionen und Argumente übernehmen. Diese Parameter sind in den [Positionsparametern](#) gespeichert und können so innerhalb des Scripts angesprochen und z. B. mit einer `case`-Konstruktion verarbeitet werden.

Wenn die Kommandozeile den konventionellen Regeln entsprechend aufgebaut ist, kann sie einfacher mit der `getopts`-Shellfunktion auseinander genommen werden ([siehe Regeln](#)).

In der Zeichenkette *Optionen* werden alle Schalter und Regler mit ihren Kennbuchstaben angegeben. Regler, die zusätzliche Argumente erhalten, werden von einem Doppelpunkt gefolgt.

Immer dann, wenn `getopts` aufgerufen wird, gibt es eine Option in der beim Aufruf bezeichneten *Variable* zurück. Wenn diese *Variable* nicht existiert, wird sie erzeugt. Wenn die Option ein Argument erwartet (also mit einem Doppelpunkt markiert ist), wird dieses Argument in der Shellvariablen `OPTARG` zurückgegeben.

Bei jedem Aufruf von `getopts` wird der Zeiger in der Shellvariablen `OPTIND` erhöht, damit bei einem weiteren Aufruf automatisch die nächste Option eingelesen wird.

Der `OPTIND` wird beim Start der Shell mit 1 initialisiert. Wenn eine Kommandozeile mehrfach eingelesen werden soll, muß der Index manuell zurückgesetzt werden.

Wenn nicht ausdrücklich eine *Argument*-Zeichenkette beim Aufruf von `getopts` übergeben wird, nimmt das Shellkommando die Positionsparameter, also die Argumente von der Kommandozeile des Scripts.

Der Status von `getopts` ist 0 (wahr), wenn eine Option gefunden wurde und falsch, wenn das Ende der Kommandozeile bzw. der Argumente erreicht ist oder wenn ein Fehler aufgetreten ist.

Die Ausgabe von Fehlermeldungen in den Fehlerkanal kann durch Plazieren eines Doppelpunktes als erstes Zeichen der Optionen-Zeichenkette oder durch Belegung der Shellvariablen `OPTERR=0` unterdrückt werden.

Zur weiteren Fehlerbehandlung kann bei der „stillen“ Variante in der *Variablen* ein symbolischer Fehlercode und in der Shellvariablen `OPTARG` das zuletzt gelesene Token gefunden werden.

## hash

Syntax: **hash** [-r] [-p *Pfadname*] [*Name*]

Die Shell unterhält eine Hashtabelle, in der alle seit dem Start der Shell aufgerufenen (externen) Programme mit komplettem Pfadnamen gespeichert werden. Das beschleunigt jeden weiteren Aufruf eines solchen Programms, weil nicht erst auf dem Pfad danach gesucht werden muß. Wenn das Shellkommando `hash` mit dem *Namen* eines externen Programms aufgerufen wird, fügt es diesen Namen (mit Pfad) in die Hashtabelle ein.

Wenn die Option `-r` gesetzt ist, wird die Hashtabelle gelöscht. Diese Option kann notwendig sein, wenn eine Binärdatei gelöscht oder verschoben worden ist. Wenn kein Argument angegeben ist, wird der Inhalt der Hashtabelle ausgegeben.

Mit der Option ``-p'` kann ein *Pfadname* zum angegebenen *Namen* übergeben werden, der dann ohne weitere Suche als Hascheintrag für diesen Namen gespeichert wird.

## help

Syntax: **help** [*Shellkommando*]

Das `help`-Shellkommando zeigt einen kurzen Hilfstext zu dem angegebenen Shellkommando oder, wenn kein Kommando angegeben ist, eine Übersicht aller Shellkommandos, zu denen Hilfstexte verfügbar sind.

## history

Syntax: **history** [`-c`] [*n*]

**history** [`-anrw`] [*Datei*]

`-a` [*Datei*]

veranlaßt das Sichern der in der aktuellen Sitzung neu entstandenen History-Zeilen in das History-File bzw. in die angegebene *Datei*

`-c`

löscht alle Einträge in der History

`-n` [*Datei*]

veranlaßt das Lesen neuer Einträge aus dem History-File bzw. aus der angegebenen *Datei*

`-p` *Muster*

führt eine [History-Expansion](#) mit dem angegebenen *Muster* durch

`-r` [*Datei*]

ersetzt den Inhalt der History durch den Inhalt der *Datei*

`-s` *Eintrag*

hängt den angegebenen *Eintrag* an das Ende der History an

Anstatt den *Dateinamen* für das History-File als Argument zu übergeben kann dieser auch in der Shellvariablen `HISTFILE` festgelegt werden. Voreinstellung ist `~/ .bash_history`.

## jobs

Syntax: **jobs** [`-lnprs`] [*Jobspezifikation*]

**jobs** `-x` *Kommando* [*Argument...*]

Das Shellkommando `jobs` gibt eine Liste der aktuellen Jobs aus. In dieser Liste steht neben der Jobnummer zu jedem Job der Kommandoname, der Status und eine Markierung ``+'` für den 'aktuellen Job' und ``-'` für den vorhergehenden aktuellen [Job](#).

Wenn die Option ``-l'` gesetzt ist, wird zusätzlich die Prozeßnummer zu jedem Job ausgegeben. Mit der Option ``-p'` wird nur die Prozeßnummer ausgegeben. Mit der Option ``-n'` werden nur die Jobs angezeigt, die ihren Status seit der letzten Anzeige geändert haben. Wenn eine Jobspezifikation

angegeben ist, werden nur die Daten zu diesem Job angezeigt.

Mit den Optionen ``-r'` und ``-s'` wird die Auflistung auf laufende (running) oder angehaltene (stopped) Jobs beschränkt.

Durch Angabe der Option ``-x'` wird das angegebene *Kommando* mit den *Argumenten* ausgeführt. Alle in den Argumenten auftauchenden Jobspezifikationen werden dabei durch die entsprechenden Prozeßnummern ersetzt und können so vom *Kommando* bearbeitet werden..

## kill

Syntax: **kill** [-s *SignalSpezifikation*] [-n *Signalnummer* | -*SignalSpezifikation* [*Prozeßnummer* | *Jobspezifikation*]

**kill** -l [*Signalnummer* | -*SignalSpezifikation*]

Das Shellkommando `kill` sendet ein *Signal* an den durch die *Prozeßnummer* oder die *Jobspezifikation* identifizierten Prozeß. Standardsignal ist SIGTERM (15) zum Terminieren des Prozesses. Es können aber auch beliebige andere Signale gesendet werden. Das Signal kann als Name oder als Nummer angegeben werden. Bei der Angabe des Namens kann auf den Präfix SIG verzichtet werden. Die Jobspezifikation ist [hier](#) erklärt.

Mit der Option ``-l'` werden alle möglichen Signalnamen aufgelistet.

Es gibt auch ein externes `kill`-Kommando, mit dem die gleichen Signale gesendet werden können, das aber nicht mit einer Jobspezifikation in der Kommandozeile umgehen kann.

## let

Syntax: **let** *Ausdruck* [*Ausdruck*...]

Das Shellkommando `let` berechnet jedes Argument als arithmetischen Ausdruck. Der Rückgabewert von `let` ist 1, wenn der letzte Ausdruck Null liefert; sonst ist der Status Null.

Die Syntax der Ausdrücke ist [hier](#) erklärt.

## local

Syntax: **local** [*Name*[=*Wert*]]

Das Shellkommando `local` erzeugt eine lokale Variable *Name* und weist ihr den *Wert* zu. Wenn eine lokale Variable innerhalb einer Funktion erzeugt wird, so ist sie nur innerhalb dieser Funktion und allen Unterfunktionen zugänglich. Außerhalb von Funktionen hat die Shellfunktion `local` keine Bedeutung.

Wenn kein Name angegeben ist, werden alle lokalen Variablen angezeigt.

# logout

Syntax: **logout**

Das Shellkommando `logout` beendet eine Loginshell. Wenn die Shell als `bash` gestartet wurde, wird dabei das Shellsript `~/ .bash_logout` abgearbeitet.

# popd

Syntax: **popd** [-n] [+|-n]

Das `popd`-Shellkommando löscht einen Verzeichnisnamen vom Verzeichnisstapel. Ohne Argument wird das erste (oberste) Verzeichnis vom Stapel geholt und mit `cd` ein Verzeichniswechsel dorthin ausgeführt. Die Option `-n` unterdrückt die Ausführung des Verzeichniswechsels, das Verzeichnis wird aus dem Stapel ohne weitere Konsequenz entfernt.

Durch Angabe der Option `+/-n` kann ein bestimmtes Verzeichnis aus dem Stapel gelöscht werden. Dabei ist die Zahl `n` die Position im Stapel, beginnend mit Null, und das Vorzeichen gibt an, ob das Verzeichnis vom „Anfang“ (links in der Liste von `dirs`, mit `+`) oder vom „Ende“ (`-`) aus gezählt werden soll.

Nach jedem erfolgreichen **popd** wird automatisch die Liste aller Verzeichnisse im Stapel wie vom Kommando `dirs` ausgegeben.

# pushd

Syntax: **pushd** [-n] *Verzeichnis*

**pushd** [-n] [+|-n]

Das Shellkommando `pushd` legt das *Verzeichnis* als oberstes auf dem Verzeichnisstapel ab oder rotiert den Verzeichnisstapel um die angegebenen Positionen. Die Angabe der Option `-n` unterdrückt dabei die Ausführung des sonst üblichen Verzeichniswechsels.

Nach einem erfolgreichen Verzeichniswechsel wird automatisch die Liste aller Verzeichnisse im Stapel wie vom Kommando `dirs` ausgegeben.

# pwd

Syntax: **pwd** [-LP]

Das Shellkommando `pwd` gibt den Pfadnamen des aktuellen Verzeichnisses aus. Wenn die Option `-P` angegeben ist, wird der „physische“ Pfadname angezeigt, in dem keine symbolischen Links enthalten sind. Dieses Verhalten kann durch die `set`-Option `physical` voreingestellt werden. In diesem Fall sollte durch die Option `-L` die sonst übliche Anzeige des logischen Pfades durchgesetzt werden, in der aktuellen Version 2.01 funktioniert das jedoch nicht.

## read

Syntax: **read** [-er] [-a *Arrayname*] [-p *Prompt*] [*Variablenname*...]

Die Shellfunktion `read` liest eine Zeile von der Standardeingabe und weist die (durch die IFS getrennten) Wörter den benannten Shellvariablen zu. Wenn mehr Wörter in der Zeile stehen als Variablennamen angegeben sind, werden die verbleibenden Wörter alle in der zuletzt benannten Variablen gespeichert.

-e

ermöglicht das Editieren der Eingabe mit den Editorfunktionen von `readline`

-r

ein durch ein ``\`` eingeleitetes Zeilenende wird als Teil der Eingabe in einer Variablen abgespeichert

-a *Array*

weist die eingelesenen Werte der angegebenen Array-Variablen zu; die Variable wird vor der Belegung gelöscht und von Index 0 beginnend aufgefüllt

-p *Prompt*

wenn die Eingabe von einem Terminal stattfindet wird der angegebene *Prompt* als Eingabeaufforderung angezeigt

Wenn kein Name für die Variable angegeben ist, unter dem die Eingabe gespeichert werden soll, so wird automatisch die Shellvariable `REPLY` (Antwort) benutzt.

## readonly

Syntax: **readonly** [-afnp] [*Name*]

Die Shellfunktion `readonly` gibt Variablen (oder Scriptfunktionen mit der Option ``-f'`) den „nur Lesen“-Status. Solche Variable können nicht gelöscht oder verändert werden.

Wenn kein Name angegeben ist oder die Option ``-p'` gesetzt ist, werden alle Variablen mit „nur Lesen“-Status angezeigt.

Mit der Option ``-a'` kann die Ausgabe der Variablenliste auf Arrays beschränkt werden.

Mit der Option ``-n'` können bis zur Version 1.14.2 Variable mit „nur Lesen“-Status wieder beschreibbar gemacht werden. Die einzige Ausnahme sind die reale und die effektive User-ID, die nicht direkt verändert werden können.

## return

Syntax: **return** [*n*]

Die `return`-Shellfunktion hat nur innerhalb einer Scriptfunktion eine Bedeutung und veranlaßt dort, die Funktion mit dem angegebenen Rückgabewert zu verlassen.

Wenn kein Rückgabewert angegeben ist, wird der Status des zuletzt ausgeführten Kommandos weitergereicht.



# set

Syntax: **set** [-aBbCefHhknotuvx] [-o *Attribut*] [*Positionsparameter...*]

Mit dem Shellkommando **set** werden die Attribute der Shell zur Laufzeit verändert. Die Schalter können auch beim Aufruf der Shell auf der Kommandozeile übergeben werden.

Ohne Argumente wird eine Liste aller Umgebungsvariablen ausgegeben. Das Format ist zum Wiedereinlesen geeignet.

- a  
veranlaßt die Shell, alle neu erzeugten oder veränderten Variablen automatisch zu exportieren
- B  
diese Option schaltet die [Klammererweiterung](#) ein (Voreinstellung)
- b  
zeigt die Beendigung eines Jobs sofort an, ohne auf die nächste Eingabeaufforderung zu warten
- C  
verbietet das Überschreiben existierender Dateien durch Ausgabeumlenkung (wie `noclobber`)
- e  
beendet die Shell sofort, wenn ein Kommando nicht den Rückgabewert Null liefert; bei zusammengesetzten Kommandos ist der Status nach der kompletten Bearbeitung entscheidend, nicht das Ergebnis eines einfachen Kommandos
- f  
unterdrückt die Pfadnamenerweiterung
- H  
ermöglicht den Bezug auf Zeilen im Kommandozeilenspeicher mit dem ``!'` wie in der `csch`
- h  
veranlaßt die Shell die Pfadnamen der externen Programme bei ihrem Aufruf zur Beschleunigung wiederholter Ausführung in einer Hashtabelle zu speichern (Voreinstellung)
- k  
veranlaßt die Shell, alle beim Aufruf eines Kommandos auf der Kommandozeile übergebenen Zuweisungen in die Umgebung zu exportieren, nicht nur die dem Programmnamen Vorhergehenden
- m  
ermöglicht die Benutzung der Job-Kontrollfunktionen
- n  
liest Kommandos, ohne sie auszuführen; diese Option funktioniert nicht in interaktiven Shells und dient zum Testen von Shellscripts
- o **Shelloption**  
setzt die angegebene *Shelloption*. Diese Einstellungen können auch mit dem Shellkommando `shopt` vorgenommen werden. Folgende Einstellungen sind möglich:

allexport

das gleiche wie die Option -a

braceexpand

in geschweiften Klammern eingeschlossene, durch Kommata getrennte Listen von Wortteilen in der Kommandozeile werden durch mehrere Wörter mit je einem eingefügten Wortteil ersetzt (wie in der C-Shell)

emacs

schaltet den Kommandozeileneditor in den emacs-Stil

errexist

das gleiche wie die Option -e

hashall

das gleiche wie die Option -h

histexpand

das gleiche wie die Option -H

history

ermöglicht die Verwendung der History-Funktion (Voreinstellung bei interaktiven Shells)

ignoreeof

unterdrückt das Verlassen der Shell beim Lesen von EOF mit der Wirkung von IGNOREEOF=10

keyword

das gleiche wie die Option -k

monitor

das gleiche wie die Option -m

noclobber

verbietet das Überschreiben existierender Dateien durch Ausgabeumleitung wie -C

noexec

das gleiche wie die Option -n

noglob

das gleiche wie die Option -f

notify

das gleiche wie die Option -b

nounset

das gleiche wie die Option -u

oncmd

das gleiche wie die Option -t

physical

das gleiche wie die Option -P

posix

schränkt alle Funktionen der Shell so ein, daß sie dem POSIX-Standard genügen

privileged

das gleiche wie die Option -p

verbose

das gleiche wie die Option -v

vi

schaltet den Kommandozeileneditor in den vi-Stil

xtrace

das gleiche wie die Option -x

-P

veranlaßt die Verwendung des "physischen" Pfades beim Wechsel in Verzeichnisse, deren Pfad symbolische Links enthält

-p

(privileged) veranlaßt die Shell, beim Start die ENV- Initialisierungsdatei und die aus der Umgebung geerbten Funktionen zu ignorieren; dieser Schalter wird automatisch gesetzt, wenn die effektive User-ID nicht mit der realen übereinstimmt; beim Zurücksetzen des Schalters wird die effektive User-ID sofort mit der realen gleichgesetzt

-t

beendet die Shell sofort nach der Ausführung eines einzigen Kommandos

-u

erzeugt eine Fehlermeldung für jede leere (ungesetzte) Variable, die erweitert werden soll

-v

gibt jede Kommandozeile so aus, wie sie gelesen wurde

-x

gibt nach der Erweiterung jedes einfachen Kommandos den Inhalt der Shellvariablen PS4, gefolgt von dem erweiterten Kommando mit allen Argumenten, aus

-

setzt die Positionsparameter auf die der Option folgenden Werte, auch wenn einer der Werte mit einem '-' beginnt; wenn keine Werte folgen, werden die Positionsparameter gelöscht

-

setzt die Positionsparameter auf die der Option folgenden Werte; wenn keine Werte folgen, bleiben die Positionsparameter unverändert

Wenn anstelle des '-' bei den Optionen ein '+' gesetzt ist, wird die entsprechende Option abgeschaltet. Alle hier aufgeführten Optionen können auch beim Aufruf der Shell in der Kommandozeile gesetzt werden. Die aktuell gesetzten Optionen können mit der Shellvariablen '-' angezeigt werden (echo\$-).

Wenn keine Optionen angegeben sind, werden alle Shellvariablen mit ihren Werten angezeigt.

## shift

Syntax: **shift** [*n*]

Die shift-Shellfunktion verschiebt (shiftet) die Positionsparameter um *n* Stellen nach links. Die

herausgeschobenen Parameter sind verloren. Wenn keine Anzahl angegeben ist, wird um eine Stelle geshiftet.

## shopt

Syntax: **shopt** [-pqsu] [-o] [*Shelloption*]

Das Shellkommando `shopt` dient speziell zur Veränderung unterschiedlicher Laufzeitoptionen der Shell. Zusätzlich zu den auch vom Shellkommando `set` verwalteten Optionen sind durch `shopt` weitere Einstellungen möglich.

- o  
ermöglicht die Veränderung der von `set` verwalteten Shellattribute
- p  
gibt eine Liste der Shelloptionen mit ihren Einstellungen aus
- q  
(quiet) unterdrückt die Bildschirmausgabe; der Statuswert gibt an, ob eine Shelloption gesetzt ist oder nicht
- s  
veranlaßt das Setzen der angegebenen Option bzw. das Auflisten aller gesetzten Optionen
- u  
veranlaßt das Löschen der angegebenen Option bzw. das Auflisten aller ungesetzten Optionen

Die speziell von `shopt` verwalteten Shelloptionen sind:

`cdable_vars`

Wenn diese Option gesetzt ist, kann dem Shellkommando `cd` eine Shellvariable anstelle eines Verzeichnisnamen übergeben werden, aus der der Name des Zielverzeichnisses gelesen wird.

`cdspell`

Durch Setzen dieser Shelloption wird eine automatische Korrekturfunktion aktiviert, die kleine Tippfehler bei der Angabe des Zielverzeichnisses für `cd` ausgleicht.

`checkhash`

Wenn diese Option gesetzt ist, überprüft die `bash` vor dem öffnen einer in der Hashtabelle gespeicherten Programmdatei, ob dieses Programm noch an der gleichen Stelle zu finden ist. Sollte das Programm nicht mehr vorhanden sein, wird eine normale Pfadsuche durchgeführt.

`checkwinsize`

Wenn diese Option gesetzt ist, überprüft die `bash` nach jeder Kommandoausführung die Fenstergröße und paßt die Umgebungsvariablen `LINES` und `COLUMNS` an, wenn das erforderlich ist.

`cmdhist`

Durch Setzen dieser Option wird die `bash` veranlaßt, mehrzeilige Kommandos in einer einzigen Historyzeile zu speichern um die Wiederholung bzw. das Editieren dieser zusammenhängenden Befehle zu erleichtern.

`dotglob`

Wenn diese Option gesetzt ist, werden Dateinamen, die mit einem Punkt beginnen, in die Dateinamenerweiterung mit einbezogen.

#### `execfail`

Durch Setzen dieser Option wird verhindert, daß eine nicht-interaktive Shell nach einem Fehlgeschlagenen `exec` automatisch beendet wird. Interaktive Shells zeigen dieses Verhalten als Voreinstellung und können umgekehrt durch das Shellattribut `errexit` dazu gebracht werden, nach fehlgeschlagenem `exec` zu terminieren.

#### `expand_aliases`

Diese Option ist standardmäßig bei allen interaktiven Shells gesetzt und ermöglicht die Alias-Expansion.

#### `histappend`

Wenn diese Option gesetzt ist, wird beim Verlassen der Shell der Inhalt der History an die History-Datei angehängt anstatt sie zu überschreiben.

#### `histreedit`

Wenn diese Option gesetzt ist und die `readline`-Funktionen aktiv sind, wird die Kommandozeile nach einer fehlgeschlagenen History-Substitution nochmal im Kommandozeileneditor vorgelegt.

#### `histverify`

Nach dem Setzen dieser Option wird ein Kommando nach der History-Substitution nicht direkt ausgeführt, sondern in den Kommandozeileneditor geladen und zur interaktiven Bearbeitung angezeigt.

#### `hostcomplete`

Solange diese Option aktiv ist, wird das auf das Zeichen `@` folgenden Wort bei der automatischen Kommandozeilenerweiterung als Hostname interpretiert und anhand der Daten aus `/etc/hosts` nach Möglichkeit ergänzt.

#### `interactive_comments`

Diese Option veranlaßt die `bash`, alle auf das Kommentarzeichen `#` folgenden Worte auch auf der Kommandozeile einer interaktiven Shell als Kommentar zu behandeln.

#### `lithist`

Zusammen mit der Option `cmdhist` werden zusammenhängende Zeilen eines Kommandos in der History-Datei in mehreren Zeilen abgespeichert. Normalerweise werden die Teile eines mehrzeiligen Kommandos durch Semikolon getrennt gespeichert.

#### `mailwarn`

Wenn diese Option gesetzt ist, wird eine Warnung ausgegeben, wenn der Mail-Folder von einem anderen Account zum Lesen geöffnet wurde.

#### `nullglob`

Durch Setzen dieser Option wird bei der Pfadnamenerweiterung das Ersetzen eines Musters durch Leerstrings erlaubt. In der Voreinstellung bleiben Muster, auf die kein Name paßt, unverändert in der Kommandozeile erhalten.

#### `promptvars`

Solange diese Option gesetzt bleibt, werden im Prompt-String enthaltene Variable vor der Ausgabe expandiert.

#### `shift_verbose`

Wenn diese Option gesetzt ist, gibt das Shellkommando `shift` eine Warnung aus, wenn versucht wird, über den letzten Positionsparameter hinauszugehen.

`sourcepath`

Durch Setzen dieser Option wird die `bash` veranlaßt, die mit dem Shellkommando `source` einzulesenden Shellscrip­ts in den Verzeichnissen vom `PATH` zu suchen.

## source

Syntax: **source** *Datei*

Die `source`-Shellfunktion läßt die Shell das Shellscrip­*t* *Datei* abarbeiten. Dabei wird kein neuer Shell-Prozeß gestartet, sondern der Inhalt der Scriptdatei in den Eingabekanal der aktuellen Shell eingespeist. Diese Funktion kann auch durch einen einzelnen Punkt am Anfang der Kommandozeile ausgelöst werden.

## suspend

Syntax: **suspend** [ `-f` ]

Die `suspend`-Shellfunktion veranlaßt die Shell, auf das Signal `SIGCONT` zu warten. Wenn die Option ``-f'` gesetzt ist, kann auch eine Loginshell mit dieser Funktion angehalten werden.

## test

Syntax: **test** *Ausdruck*

[*Ausdruck*]

Die Shellfunktion `test` bewertet den Ausdruck und liefert Null, wenn der Ausdruck wahr (!) ist und Eins, wenn er falsch ist. Dieser Unterschied zu der gängigen Definition der Wahrheitswerte in C ist für die Shellprogrammierung normal! (Siehe auch im Abschnitt über den [Status der Kommandos](#))

Ein Ausdruck kann einen einstelligen (unären) oder einen zweistelligen (binären) Operator enthalten. Einstellige Operatoren benötigen ein einziges Argument, zweistellige Operatoren stehen zwischen zwei Argumenten. Unäre Operatoren dienen oft zum Ermitteln des Zustandes einer Datei.

Außerdem können mehrere Ausdrücke noch durch spezielle Operatoren verknüpft werden.

Folgende Operationen können ausgeführt werden:

`-b` *Datei*

ist wahr, wenn die *Datei* ein Blockdevice ist

`-c` *Datei*

ist wahr, wenn die *Datei* ein Zeichendev­ice ist

`-d` *Datei*

ist wahr, wenn die *Datei* ein Verzeichnis ist

`-e` *Datei*

ist wahr, wenn die *Datei* existiert

-f ***Datei***

ist wahr, wenn die *Datei* eine einfache Datei (plain file) ist

-g ***Datei***

ist wahr, wenn bei der *Datei* das SGID Bit gesetzt ist

-k ***Datei***

ist wahr, wenn bei der *Datei* das „sticky“-Bit gesetzt ist

-L ***Datei***

ist wahr, wenn die *Datei* ein symbolischer Link ist

-p ***Datei***

ist wahr, wenn die *Datei* eine benannte Pipeline (Named Pipe) ist

-r ***Datei***

ist wahr, wenn die *Datei* existiert und lesbar ist

-s ***Datei***

ist wahr, wenn die *Datei* existiert und größer als Null Bytes ist

-S ***Datei***

ist wahr, wenn die *Datei* ein „Socket“ ist

-t ***Dateinummer***

ist wahr, wenn die Datei mit der *Dateinummer* für ein Terminal geöffnet ist. Wenn keine Nummer angegeben ist, wird Nummer 1 (Standardausgabe) angenommen

-u ***Datei***

ist wahr, wenn die *Datei* existiert und das SUID Bit gesetzt ist

-w ***Datei***

ist wahr, wenn die *Datei* existiert und beschreibbar ist

-x ***Datei***

ist wahr, wenn die *Datei* existiert und ausführbar ist

-O ***Datei***

ist wahr, wenn die *Datei* existiert und im Eigentum des Anwenders ist, unter dessen UID das `test`-Shellkommando läuft

-G ***Datei***

ist wahr, wenn die *Datei* existiert und im Eigentum des Benutzers ist, unter dessen GID das `test`-Shellkommando läuft

***Datei1* -nt *Datei2***

(newer than) ist wahr, wenn die *Datei1* neuer ist als die *Datei2*

***Datei1* -ot *Datei2***

(older than) ist wahr, wenn die *Datei1* älter ist als die *Datei2*

***Datei1* -ef *Datei2***

(equal to file) ist wahr, wenn *Datei1* und *Datei2* die gleiche Inode auf dem gleichen Device belegen

### -o *Attribut*

ist wahr, wenn das angegebene [Shellattribut](#) gesetzt ist

### -z *Zeichenkette*

ist wahr, wenn die Länge der *Zeichenkette* Null ist

### -n *Zeichenkette*

ist wahr, wenn die Länge der *Zeichenkette* nicht Null ist

### *Zeichenkette*

ist auch wahr, wenn die Länge der *Zeichenkette* nicht Null ist, es wird also nicht der (eventuell numerische) Inhalt der Variablen getestet

### *Zeichenkette1* = *Zeichenkette2*

ist wahr, wenn die Zeichenketten gleich sind

### *Zeichenkette1* != *Zeichenkette2*

ist wahr, wenn die Zeichenketten nicht gleich sind

### *Zeichenkette1* < *Zeichenkette2*

ist wahr, wenn die *Zeichenkette1* lexikalisch vor der zweiten Zeichenkette eingeordnet wird

### *Zeichenkette1* > *Zeichenkette2*

ist wahr, wenn die *Zeichenkette1* lexikalisch nach der zweiten Zeichenkette eingeordnet wird

### ! *Ausdruck*

ist wahr, wenn der Ausdruck falsch ist

### *Ausdruck1* -a *Ausdruck2*

ist wahr, wenn *Ausdruck1* UND *Ausdruck2* wahr sind

### *Ausdruck1* -o *Ausdruck2*

ist wahr, wenn *Ausdruck1* ODER *Ausdruck2* wahr ist

### *Argument1* OP *Argument2*

OP steht hier für einen der arithmetischen Vergleich -eq, -ne, -lt, -le, -gt und -ge (gleich, ungleich, kleiner, kleinergleich, größer, größergleich) Der Ausdruck ist wahr, wenn die Relation von *Ausdruck1* und *Ausdruck2* stimmt.

## time

Syntax: **time** [-p] *Kommando*

**time** führt das angegebene *Kommando* aus, ermittelt die Zeit, die dafür benötigt wurde und gibt diese Daten aus. Die Zeit ist aufgeteilt in **real** (die tatsächlich vergangene Zeit) **user** (die Rechenzeit im Usermodus) und **sys** (Rechenzeit im Kernelmodus).

Mit der Option -p werden die Zeitangaben POSIX-konform formatiert. Durch die Shellvariable [TIMEFORMAT](#) kann das Ausgabeformat für die Zeitangabe weiter angepaßt werden.

Das seit Version 2.0 von der **bash** intern angebotene **time** spielt eine Sonderrolle in dieser Auflistung von Shellkommandos. Es bildet im Prinzip das externe Programm **time** nach, wird also wie ein Shellkommando benutzt. Das interne **time** der **bash** wird jedoch nicht als Shellkommando behandelt, sondern als Schlüsselwort. Damit hat es den gleichen Status wie **if...fi** oder **while...do...done**.



Insbesondere läßt sich das interne `time` nicht durch `enable -n time` abschalten.

Der Vorteil des internen `time` besteht darin, daß es die Ausführungszeit einer kompletten [Pipeline](#) messen kann. Das folgende Beispiel zeigt den Unterschied:

```
$ /usr/bin/time ls -l /usr/bin | sort
0.17user 0.04system 0:00.27elapsed 77%CPU (0avgtext+0avgdata 0maxresident)
0inputs+0outputs (0major+0minor)pagefaults 0swaps
-r-sr-xr-x  1 root    bin          13234 Mar 21  1997 passwd
-r-xr-xr-x  1 bin     root         3060 Jul 30 17:09 uptime
-r-xr-xr-x  1 bin     root         4912 Jul 30 17:09 watch
...
$
```

Das externe Kommando führt dazu, daß ein Listing von `/usr/bin` erzeugt und dabei die Ausführungsdauer für dieses eine Kommando ermittelt wird. Diese Zeit wird in den Fehlerkanal geschrieben und erscheint auf dem Bildschirm, noch während `sort` das Verzeichnislisting sortiert. Wenn als Ergebnis die gesamte Ausführungsdauer der Pipeline mit `ls` und `sort` gefragt ist, kann dieser Wert nur vom internen `time` ermittelt werden.

Weil die Pipeline auf der syntaktischen Ebene der Shell zusammengestellt wird, muß `time` auf der gleichen Ebene angesiedelt sein.

## times

Syntax: **times**

Das Shellkommando `times` gibt die verbrauchte Benutzer- und Systemzeit jeweils für die Shell und für die von der Shell aus gestarteten Prozesse an.

## trap

Syntax: **trap** [-lp] [*Kommando*] [*Signalspezifikation*]

Die Shellfunktion `trap` fängt das angegebene *Signal* ab und führt das *Kommando* aus. Wenn kein *Signal* benannt ist, werden alle Signale zurückgesetzt. Wenn als *Kommando* die leere Zeichenkette angegeben ist, wird das damit angegebene *Signal* von der Shell und von allen Kommandos, die von dieser Shell ausgeführt werden, ignoriert.

Das *Signal* kann entweder als Zahl oder mit seinem Namen angegeben werden. Eine Liste aller möglichen Signale kann vom Shellkommando `trap` mit der Option `-l` ausgegeben werden.

Wenn das *Signal* `EXIT` (0) angegeben ist, wird das *Kommando* als letztes vor der Beendigung der Shell ausgeführt.

Wenn keine Argumente oder nur die Option `-p` angegeben sind, wird eine Liste aller „getrapten“ Signale und der damit verbundenen Kommandos ausgegeben.

Durch die spezielle *Signalspezifikation* `DEBUG` wird erreicht, daß das mit `trap` festgelegte *Kommando* nach jedem einfachen Kommando ausgeführt wird.

# type

Syntax: **type** [-atp] [-all] [-type | -path] [*Name*]

Die Shellfunktion `type` gibt an, wie der angegebene *Name* von der Shell interpretiert würde, wenn er in der Kommandozeile an der Position eines Kommandos stünde (alias, Scriptfunktion, Shellfunktion (builtin), Datei). Wenn der Name nicht gefunden wird, gibt `type` nichts aus.

Mit der Option `-type` wird nur ein Wort für den Kommandotyp entsprechend den oben genannten Möglichkeiten ausgegeben.

Wenn die Option `-path` benutzt wird, gibt die Shellfunktion den kompletten Pfadnamen des benannten Kommandos aus. Wenn es kein externes Kommando mit dem Namen gibt, wird nichts ausgegeben.

Die Option `-all` veranlaßt die Shellfunktion nicht nur die erste Fundstelle eines passenden Kommandos anzuzeigen, sondern alle möglichen. Mit der Option `-path` kann dabei zusätzlich die Ausgabe auf externe Kommandos eingeschränkt werden.

## typeset

Siehe **declare**

## ulimit

Syntax: **ulimit** [-SHacdfmnpstuv] [*Limit*]

Die Shellfunktion `ulimit` erlaubt die Kontrolle über die von der Shell und den daraus gestarteten Programmen benutzten Systemressourcen.

-S

setzt „weiche“ Grenzen; die Veränderung solcher Grenzen innerhalb des durch „harte“ Grenzen gegebenen Rahmens ist jederzeit möglich

-H

setzt „harte“ Grenzen; eine einmal gesetzte harte Grenze kann nicht nach oben erweitert werden

-a

zeigt alle eingestellten Grenzwerte an

-c

schränkt die Größe des Speicherabzugs (core) bei einem Programmabsturz ein

-d

schränkt die maximale Größe des Datensegments jedes einzelnen Prozesses ein, der von dieser Shell aus gestartet wird; der Versuch eines Prozesses über diese Grenze hinaus Speicher zu allozieren ist erfolglos

-f

verbietet dem Anwender, Dateien über einer bestimmten Größe zu erzeugen; ist nur im Extended-2 Filesystem implementiert

-n

schränkt die maximale Anzahl offener Dateien jedes einzelnen von dieser Shell gestarteten Prozesses ein

-m

nicht implementiert; ist dafür vorgesehen, die Größe des residenten (nicht auszulagernden) Teiles der Prozesse einzuschränken

-p

zeigt die Größe des Pipeline-Puffers (in 512 Byte Blöcken) an; dieser Wert kann nicht verändert werden

-s

schränkt den Stapelspeicher (Stack) jedes einzelnen von dieser Shell aus gestarteten Prozesses auf eine bestimmte Größe ein

-t

schränkt die verfügbare CPU-Zeit (User und System) jedes einzelnen Prozesses auf die angegebene Anzahl Sekunden ein

-u

schränkt die Anzahl der Prozesse je Benutzer ein; dabei werden auch die von anderen Shells gestarteten Prozesse des selben Benutzers mitgezählt


-v

zeigt das Gesamtlimit des virtuellen Speichers für jeden aus dieser Shell gestarteten Prozeß an

Die Grenzen werden in Kilobytes angegeben, wenn oben keine andere Einheit genannt ist. Wenn beim Aufruf keine Grenze bestimmt wird, gibt `ulimit` die aktuelle Grenze an.

## umask

Syntax: **umask** [-S] [*Modus*]

Die Shellfunktion `umask` setzt die Maske, mit der die Zugriffsrechte auf Dateien und Verzeichnisse unmittelbar nach ihrer Erzeugung durch einen von dieser Shell kontrollierten Prozeß  bestimmt werden. Die in der Maske gesetzten Bits werden bei den Zugriffsrechten für eine neue Datei (oder ein neues Verzeichnis) gelöscht (sie werden maskiert).

Die Maske kann als Oktalzahl oder in der beim Kommando [chmod](#) angegebenen Form angegeben werden. Wenn kein Wert angegeben ist, wird die aktuelle Maske angezeigt. Wenn die Option `-S` gesetzt ist, wird die aktuelle Maske in symbolischer Form ausgegeben.

Die Maske `'022'` verbietet beispielsweise allen Benutzern, außer dem Eigentümer selbst, das Schreiben in eine neu angelegte Datei oder ein Verzeichnis.

Die Abbildung [3.1](#) veranschaulicht die Funktionsweise von `umask`:

-	r	w	x	r	w	x	r	w	x
	↓	↓	↓	↓	↓	↓	↓	↓	↓
-	r	w	x	r	-	x	r	-	x

**Abbildung:** Maskierung der Zugriffsrechte durch umask

Bei der Erzeugung einer Datei wird die Funktion `creat(2)` mit einem Wert für die Permissions aufgerufen, beispielsweise 0777. Durch die `umask` werden die Schreibrechte für Gruppe und Andere gelöscht. Die übrigen Rechte kommen unverändert durch die Maske und erscheinen in der I-Node der frisch erzeugten Datei.

## unalias

Syntax: **unalias** [-a] [*Name...*]

Die Shellfunktion `unalias` hebt ein durch das `alias`-Shellkommando gesetztes Synonym für ein Kommando wieder auf.

Mit der Option `-a` werden alle Synonyme gelöscht.

## unset

Syntax: **unset** [-fv] [*Name...*]

Mit der Shellfunktion `unset` werden Shellvariable oder Shellfunktionen aus dem Speicher entfernt. Mit der Option `-f` wird die bezeichnete Funktion aus dem Speicher gelöscht, mit der Option `-v` die entsprechende Variable. Wenn keine der Optionen angegeben ist, wird zuerst versucht, eine Variable mit passendem Namen zu entfernen, und nur wenn dieser Versuch fehlschlägt, wird die entsprechende Funktion aus der Shellumgebung entfernt.

Eine Variable bzw. eine Funktion kann nur mit dem `unset`-Kommando wirklich aus dem Speicher entfernt werden. Wenn eine Variable mit der leeren Zeichenkette ``` ``` belegt ist, gilt sie weiterhin als gesetzt.

Die Shellvariablen `PATH`, `IFS`, `PPID`, `PS1`, `PS2`, `UID` und `EUID` können nicht aus dem Arbeitsspeicher der Shell entfernt werden.

## wait

Syntax: **wait** [*Jobspezifikation* | *Prozeßnummer*]

Die Shellfunktion `wait` wartet auf die Beendigung des durch die *Jobspezifikation* (Jobnummer oder Kommandoname) oder die Prozeßnummer angegebenen Hintergrundprozesses und gibt dessen Status aus.

Wenn kein Job spezifiziert wurde, wartet die Shellfunktion auf alle aktiven Hintergrundprozesse.

# Login- und andere Shells

Wenn die `bash` mit einem `-` als erstem Zeichen des nullten Arguments aufgerufen wird (wie es das `login`-Kommando macht) oder wenn die Shell mit der Option `-login` aufgerufen wird, arbeitet sie als Loginshell.

Bei einer interaktiven Shell ist die Standardeingabe und die Standardausgabe mit einem Terminal (bzw. der Konsole) verbunden. Wenn die Shell mit der Option `-i` gestartet wird, arbeitet sie interaktiv.

Eine Loginshell arbeitet vor der ersten Kommandozeile eine Reihe von Shellscripts zur Initialisierung ab:

- Wenn ein Shellscrip `/etc/profile` existiert (und lesbar ist), werden die darin beschriebenen Einstellungen und Kommandos ausgeführt.
- Wenn die Datei `~/ .bash_profile` existiert, werden anschließend die darin enthaltenen Einstellungen und Kommandos zusätzlich ausgeführt. Wenn diese Datei nicht existiert, wird nacheinander noch nach den Dateien `~/ .bash_login` und `~/ .profile` gesucht, die im Falle ihrer Existenz ebenfalls gelesen und abgearbeitet werden.
- Eine interaktive Shell, die keine Loginshell ist, arbeitet die Datei `~/ .bashrc` zur Initialisierung ab.
- Eine nicht-interaktive Shell schließlich benutzt zur Initialisierung die in der Shellvariablen `ENV` angegebene Datei.

Beim Verlassen der Loginshell (logout) wird die Datei `~/ .bash_logout` abgearbeitet.

## Optionen

Zusätzlich zu den beim Shellkommando [set](#) beschriebenen Optionen und Schaltern, die auch beim Aufruf der `bash` in der Kommandozeile verwendet werden können, versteht die `bash` folgende Optionen:

### `-c` *Zeichenkette*

veranlaßt die Shell, nur die Kommandos in der *Zeichenkette* zu bearbeiten und danach automatisch zu beenden

### `-i`

startet die Shell interaktiv, alle Befehle werden von der Standardeingabe gelesen

### `-s`

zwingt die Shell interaktiv zu starten, auch wenn nach den Optionen weitere Argumente folgen; dadurch können Shellvariable (Positionsvariable) über die Kommandozeile gesetzt werden

Außerdem versteht `bash` auch noch eine Reihe von Klartextoptionen. Diese Optionen müssen unbedingt vor allen einfachen Buchstabenoptionen gesetzt werden. Bei älteren Versionen der Shell (vor `bash-2.0`) werden die Klartextoptionen nur durch ein Minuszeichen eingeleitet.

### `-dump-strings`

ist äquivalent zu `-D`

### `-help`

gibt eine Übersicht zu den erlaubten Kommandozeilenoptionen aus

`-login`

veranlaßt den Start wie als Loginshell

`-noediting`

unterdrückt die Funktionen des Kommandozeileneditors (von Version 2.0 hieß diese Option `-nolineediting`)

`-noprofile`

unterdrückt beim Aufruf als Loginshell die Bearbeitung der Initialisierungsdateien `/etc/profile` und `~/ .bash_profile`

`-norc`

unterdrückt die Bearbeitung der Initialisierungsdatei `~/ .bashrc`; das ist die Voreinstellung, wenn `bash` unter dem Namen **sh** aufgerufen wird

`-posix`

startet die `bash` im POSIX-Modus

`-rcfile Datei`

verwendet die *Datei* anstelle der `~/ .bashrc` zur Initialisierung

`-restricted`

veranlaßt den Start mit eingeschränkten Shellfunktionen

`-version`

zeigt die Versionsnummer der Shell beim Start

## Argumente beim Aufruf der Shell

Wenn nach allen Optionen weitere Argumente in der Kommandozeile stehen und weder die Option `-c` noch die Option `-s` gesetzt sind, wird das erste verbleibende Argument als Dateiname interpretiert, aus dem die weiteren Kommandos gelesen werden (Shellscript). Dieser Dateiname ist dann in der Shellvariablen `0`, alle weiteren Argumente in den folgenden Positionsvariablen für die Bearbeitung im Shellscript zugänglich. Wenn alle Kommandos aus der Datei abgearbeitet sind, wird die `bash` automatisch beendet.

## Dateien

`/bin/bash` ist das ausführbare Programm.

`/bin/sh` ist eigentlich die Standard-Bourne-Shell. Weil die aber keine Freie Software ist, wird unter Linux meistens ein Link auf `bash` gesetzt.

`/etc/profile` ist die Standardinitialisierungsdatei für die Bourne-Shell und auch für die `bash` als Loginshell. Das folgende Beispiel zeigt eine `/etc/profile`-Datei:

```
export OPENWINHOME=/usr/openwin
export DISPLAY=":0"
```

```
PATH="/bin:/usr/bin:/usr/local/bin:/usr/X386/bin:$OPENWINHOME/bin"
PS2='> '
```

```
umask 022
ulimit -c 0
```

Wenn die bash als Loginshell gestartet wird, versucht sie, eine zusätzliche Initialisierungsdatei im Heimatverzeichnis auszuführen.

Dabei wird in der Reihenfolge nach folgenden Dateien gesucht:

```
~/.bash_profile
~/.bash_login und
~/.profile.
```

Die erste existierende Datei wird ausgeführt, die weiteren Dateien werden ignoriert. Folgendes Beispiel zeigt eine .bash\_profile-Datei:

```
PATH=$PATH:/usr/TeX/bin:/usr/local/scripts:~/bin:.
declare -r PATH

CDPATH=~:/usr/src:/usr
PS1='\w\n\u \$ '
MAIL=/var/spool/mail/she
export EDITOR=/usr/bin/vi
```

```
HISTSIZE=200
HISTFILESIZE=100
history_control=ignoredups
```

```
alias home=cd
alias l='v -a'
alias term='term < /dev/cua1 > /dev/cua1 2> /dev/null &'
```

```
source .bashrc
```

~/ .bashrc ist die Initialisierungsdatei für die bash als interaktive Nicht-Loginshell. Das folgende Beispiel zeigt eine sinnvolle .bashrc-Datei:

```
if [ $WINDOWID ]; then
    TERM=xterm
    export XDVIFONTS=/usr/TeX/lib/tex/fonts/%f.%d%p
else
    TERM=con100x40
fi
```

~/ .bash\_logout wird beim Verlassen der Login-Shell ausgeführt. Diese Datei kann beispielsweise folgende Zeilen enthalten:

```
sort .bash_history > .tmp
uniq .tmp > .bash_history
rm .tmp
clear
```

~/`.bash_history` ist die Sicherungsdatei für den Kommandozeilenspeicher. Der Name und die Größe können in den entsprechenden [Shellvariablen](#) eingestellt werden.

~/`.inputrc` enthält die benutzerdefinierten Tastaturkommandobelegungen für den Kommandozeileneditor. Inhalt und Format sind [hier](#) beschrieben.

**Autoren:** Brian Fox, Free Software Foundation und  
ChetRamey, Case Western Reserve University

---

Next	Up	Previous	Contents	Index
------	----	----------	----------	-------

**Next:** [cat](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [basename](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)



**Next:** [chgrp](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [bash](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Beispiele:](#)
- 

# cat

## Funktion:

**cat** (concatenate) verkettet Dateien und schreibt sie in die Standardausgabe

## Syntax:

```
cat [-benstuvAET] [-number] [-number-nonblank] [-squeeze-blank]
[-show-nonprinting] [-show-ends] [-show-tabs] [-show-all] [Datei
...]
```

## Beschreibung:

**cat** liest beliebige Dateien und schreibt sie ohne Veränderung in die Standardausgabe. Durch Umlenkung der Ausgabe auf eine Datei ([siehe Ausgabeumleitung](#)) können so Dateien verkettet werden. Außerdem wird **cat** häufig benutzt, um Dateien an Programme zu übergeben, die nur von der Standardeingabe lesen. (Solche Programme werden im allgemeinen als Filter bezeichnet.) Für die Übergabe steht entweder die Ausgabeumlenkung der Shell mit `>' und `>>' zur Verfügung, oder die Ausgabe wird durch eine [Pipeline](#) an den Filter weitergeleitet.

## Optionen:

- b  
alle nicht leeren Zeilen erhalten eine Zeilennummer
- e  
das gleiche wie -vE
- n  
sämtliche Zeilen werden numeriert
- s

mehrere leere Zeilen in Folge werden zu einer einzigen leeren Zeile zusammengefaßt

-t

das gleiche wie -vT

-u

ohne Funktion

-v

alle Kontrollzeichen außer TAB und NEWLINE werden angezeigt

-A

das gleiche wie -vET

-E

gibt ein `\$\$' Zeichen am Ende jeder Zeile aus

-T

die TAB werden als ^I angezeigt

## Beispiele:

Mit dem cat-Kommando können Dateien ausgedruckt werden. Mit dem Kommando

```
$ cat Adressenliste > /dev/lp1
```

```
$ _
```

können Sie die Datei Adressenliste direkt der Gerätedatei /dev/lp1 zuführen, die den ersten parallelen Druckerport im System darstellt. Diese Methode ist aber sehr unelegant. Sie umgeht und blockiert den Druckerspooler lpd, der über die reine Auslieferung hinaus noch die Formatierung der Dokumente erledigen kann. Wenn ein Druckerdämon installiert ist, sollten Sie die „direkte“ Methode nur in Ausnahmefällen benutzen.

Wenn Sie zwei Dateien durch Umlenkung der Ausgabe von cat verketten wollen, müssen Sie den Standardausgabekanal immer in eine dritte Datei zielen lassen.

```
$ cat foo bar > foo
```

```
cat: foo: input file is output file
```

```
$ _
```

führt NICHT zum erwünschten Ergebnis. Die Datei `foo' wird von der Shell vor der Ausführung von cat gelöscht, um die Ausgabe des Kommandos dorthin umzulenken. cat erkennt noch, daß eine Eingabedatei mit der Ausgabedatei übereinstimmt, Ihre Daten sind aber bereits verloren.

Ein verwandtes Problem kann beim Anhängen von Daten an eine bestehende Datei entstehen. Der Befehl

```
$ cat foo bar >> foo
```

```
cat: foo: input file is output file
```

```
$ _
```

würde zu einem „Kurzschluß“ führen, weil der schreibende Dateizeiger immer dem lesenden eine Dateilänge vorauslaufen würde. Nicht unbedingt das gewünschte Ergebnis. cat erkennt diesen Fehler und verweigert die Ausführung. Die Datei foo bleibt also unverändert erhalten.

## Autor:

Torbjorn Granlund und Richard Stallman

---

[Next](#) [Up](#) [Previous](#) [Contents](#) [Index](#)

**Next:** [chgrp](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [bash](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- 

# chgrp

## Funktion:

**chgrp** (change group) ändert die Gruppenzugehörigkeit einer Datei oder eines Verzeichnisses

## Syntax:

```
chgrp [-Rcfv] [-recursive] [-show-changes] [-silent] [-quiet]  
[-verbose] Gruppe Datei ...
```

## Beschreibung:

Der Befehl **chgrp** ändert die Gruppenzugehörigkeit einer Datei oder eines Verzeichnisses. Die Benutzung von **chgrp** ist nur dem Eigentümer und der Superuserin (ruth) erlaubt. Sie können Ihre eigenen Dateien nur den Gruppen zuordnen, denen Sie selbst auch angehören.

Eine Aufstellung aller zulässigen Gruppen erhalten Sie mit dem Kommando `id -a` oder mit `groups`.

## Optionen:

-c

(changes) diese Option zeigt die Dateien an, deren Gruppe geändert wird

-f

(force) es werden keine Fehlermeldungen ausgegeben

-v

(verbose) alle Aktionen werden angezeigt

-R

(recursive) die Gruppenzugehörigkeit der Dateien in den Unterverzeichnissen wird ebenfalls geändert

## Siehe auch:

[chmod](#) und [chown](#)

## Autor:

David MacKenzie

---

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

**Next:** [chsh](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [chgrp](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Beispiel:](#)
  - [Siehe auch:](#)
- 

# chmod

## Funktion:

**chmod** (change mode) ändert die Zugriffsrechte auf Dateien und Verzeichnisse

## Syntax:

**chmod** [-Rcfv] *Modus Datei ...*

## Beschreibung:


**chmod** setzt oder ändert die Zugriffsrechte auf Dateien oder Verzeichnisse. Die Benutzung von **chmod** ist nur dem Eigentümer oder der Systemverwalterin (**ruth**) erlaubt.

Die Zugriffsrechte werden als Modus bezeichnet. Der *Modus* kann entweder als (drei- oder vierstellige) Oktalzahl oder durch Buchstabenkennungen angegeben werden. Bei Angabe als Oktalzahl legen die letzten drei Ziffern jeweils die Rechte für den Besitzer, die Gruppe und die Anderen fest. Die einzelnen Bits der Oktalziffer stehen dabei für Lesen (4), Schreiben (2) und Ausführen (1).

Wenn vier Ziffern angegeben werden, so setzt die erste Ziffer spezielle Ausführungsmodi:

Wenn das erste Bit (4) dieser Zahl gesetzt ist, wird ein Programm mit der effektiven Benutzerkennung (EUID für Effective User-ID) des Besitzers dieser Datei ausgeführt.

Wenn das zweite Bit (2) dieser Zahl gesetzt ist, wird ein Programm mit der Gruppenkennung dieser Datei anstelle der realen Gruppenkennung des aufrufenden Benutzers ausgeführt.

Das dritte Bit (1) schließlich hat unter Linux nur bei Verzeichnissen eine Bedeutung. 

Die Buchstabenkennung setzt sich aus den folgenden Teilen zusammen:

[**u****g****o****a** ...][[+|=][**rwxstugo** ... ]...][, ...]

Dabei steht **u** (user) für Besitzer, **g** (group) für Gruppe, **o** (other) für Andere und **a** (all) für Alle. Die arithmetischen Symbole **+** **-** **=** geben an, ob eine Berechtigung hinzugefügt (+), gelöscht (-) oder gesetzt

(=) werden soll. Die Berechtigungen sind **r** (read) für Lesen, **w** (write) für Schreiben, **x** (execute) für Ausführen. Die Option **s** (set user/group ID on execution) ändert die effektive Benutzerkennung bei der Programmausführung. Das SGID Bit auf einem Verzeichnis sorgt dafür, daß alle Dateien, die in diesem Verzeichnis angelegt oder dorthin kopiert werden, Eigentum der entsprechenden Gruppe sind. Die Option **t** (text) schützt die Dateien eines beschreibbaren Verzeichnisses vor Löschung durch fremde Systembenutzer. Die nachgestellten **u**, **g** und **o** schützen die entsprechenden Rechte für Besitzer, Gruppe und Andere vor Veränderung (zur Benutzung im Zusammenhang mit -a).

Die Rechte von symbolischen Links werden von chmod nicht geändert. Es gelten hier immer die Rechte der Datei, auf die der Link zeigt.

## Optionen:

-c

(changes) es werden nur die Dateien angezeigt, deren Zugriffsrechte tatsächlich verändert werden

-f

(force) Fehlermeldungen wegen fehlgeschlagener Änderungsversuche werden unterdrückt

-v

(verbose) alle Aktionen werden angezeigt

-R

(recursive) die Zugriffsrechte aller Dateien in den Unterverzeichnissen werden ebenfalls geändert

## Beispiel:

oktal	verbal	Anzeige <code>ls -l</code>	Bedeutung
640	<code>u=rw, g=r, o=</code>	<code>-rw-r-----</code>	Nur der Eigentümer kann diese Datei verändern. Die Gruppenmitglieder können die Datei lesen, alle anderen haben keinen Zugriff darauf.
644	<code>a=r, u+w</code>	<code>-rw-r--r--</code>	Alle Systembenutzer dürfen die Datei lesen, nur der Eigentümer darf sie verändern.
4750	<code>a=, u=rwx, g=rx</code>	<code>-rwxr-x---</code>	Die ausführbare Datei kann vom Eigentümer gelesen, ausgeführt und verändert werden, die Gruppe darf sie lesen und mit der UID des Eigentümers ausführen, alle anderen haben keinen Zugriff auf die Datei.
1777	<code>a=rwx</code>	<code>drwxrwxrwt</code>	Dieses Verzeichnis können alle Systembenutzer als aktuelles Verzeichnis wählen, sie können es auflisten und sie können Dateien darin anlegen. Das Löschen einer fremden Datei ist in diesem Verzeichnis nicht möglich.

**Abbildung:** Beispiele für `chown`

## Siehe auch:

[chgrp](#) und [chown](#)

## Autor:

David MacKenzie

[Next](#)
[Up](#)
[Previous](#)
[Contents](#)
[Index](#)

**Next:** [chsh](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [chgrp](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)



## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
- 

# chsh

## Funktion:

**chsh** ändert den Eintrag für die Loginshell in der Paßwortdatei.

## Syntax:

```
chsh [-s Shell] [-luv] [Benutzer]
```

## Beschreibung:

**chsh** ermöglicht es jedem eingetragenen Benutzer, seine Loginshell selbst, das heißt ohne Hilfe der Systemverwalterin, zu verändern. Die Loginshell wird im letzten Feld des Benutzereintrags in der Paßwortdatei `/etc/passwd` festgelegt. Diese Datei kann nur mit den Privilegien der Superuserin verändert werden. Um auch den anderen Anwendern das Verändern des Eintrages zu erlauben, läuft das **chsh** Programm SUID root. Das heißt, bei seiner Ausführung wird die effektive Benutzerkennung der Systemverwalterin gesetzt. Um die Systemsicherheit trotzdem zu gewährleisten, können nur Programme, die in der Datei `/etc/shells` eingetragen sind, als Loginshell benutzt werden.

Normalerweise können Sie nur Ihre eigene Loginshell ändern. Die Superuserin selbst kann das Programm aber auch für andere Benutzer anwenden, indem sie den Benutzernamen in der Kommandozeile angibt.

**-s *Shell***

bestimmt die angegebene *Shell* zur neuen Loginshell

**-l**

(list) gibt eine Liste aller zugelassenen Shells aus

**-u**

(usage) gibt eine Kurzhilfe zum Programm aus

**-v**

(version) zeigt die Version des Programms an

**Autor:**

Salvatore Valente <svalente@mit.edu>

---

Next	Up	Previous	Contents	Index
------	----	----------	----------	-------

**Next:** [cksum](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [chmod](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

Next: [cmp](#) Up: [Von GNU's, Muscheln und](#) Previous: [chsh](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Siehe auch:](#)
- 

# cksum

## Funktion:

**cksum** errechnet die CRC Prüfsumme und die Anzahl Bytes für eine Datei

## Syntax:

**cksum** [*Datei* ...]

## Beschreibung:

**cksum** errechnet die Prüfsumme für eine oder mehrere Dateien nach dem im POSIX.2 Standard festgelegten „cyclic redundancy check“ (CRC) Verfahren. Wenn keine Datei oder anstelle einer Datei '-' angegeben ist, liest **cksum** die Standardeingabe und gibt die Prüfsumme aus, nachdem die Eingabe mit CONTROL-D beendet wurde.

Die Ausgabe von **cksum** besteht aus der Prüfsumme, der Dateilänge und dem Dateinamen. Diese Prüfsummen können beispielsweise beim Verschicken per UUCP den Dateien mitgegeben werden, so daß der Empfänger die Möglichkeit hat, den vollständigen und korrekten Empfang der Dateien durch den Vergleich der Prüfsummen zu bestätigen.

Es gibt noch andere Verfahren zur Prüfsummenberechnung, die nicht dem POSIX.2 Standard entsprechen. Zwei davon werden durch das **sum** Kommando angeboten werden. Das **cksum** Programm ist nicht kompatibel zu **sum**.

## Autor:

Frank Q. Xia

## Siehe auch:

[sum](#)

---



## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- 

# cmp

## Funktion:

**cmp** (compare) vergleicht zwei Dateien byteweise

## Syntax:

```
cmp [-cls] [-show-chars] [-verbose] [-silent] [-quiet] Datei1  
[Datei2]
```

## Beschreibung:

**cmp** vergleicht zwei (binäre) Dateien und liefert die dezimale Position und die Zeilennummer des ersten Bytes, in dem sich die Dateien unterscheiden. Wenn Sie anstelle eines der beiden Dateinamen ein Minuszeichen '-' angeben, liest das **cmp**-Kommando die Vergleichsdaten von der Standardeingabe. Wird nur eine Datei benannt, so wird anstelle der zweiten ebenfalls von der Standardeingabe gelesen.

## Optionen:

-c

(character) gibt die abweichenden Zeichen aus

-l

(list) gibt die Position und den oktalen Wert aller differierenden Zeichen in einer Liste aus

-s

(silent) gibt nichts auf die Standardausgabe; der Status ist 0 (wahr), wenn die Dateien übereinstimmen und 1 (falsch), wenn sie sich unterscheiden

## Siehe auch:

`diff`(info) und [comm](#)

## Autor:

Torbjorn Granlund und David MacKenzie

---

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

**Next:** [compress](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [cmp](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- 

# comm

## Funktion:

**comm** (common) vergleicht zwei sortierte Dateien

## Syntax:

```
comm [-{1,2,3}] Datei1 Datei2
```

## Beschreibung:

**comm** vergleicht zwei sortierte Dateien und gibt die gemeinsamen und die verschiedenen Zeilen jeweils in Spalten aus, indem die zweite und dritte Spalte jeweils von einem bzw. zwei TAB angeführt wird.

Die erste Spalte enthält die Zeilen, die nur in Datei1 enthalten sind. Die zweite Spalte enthält die Zeilen, die nur in der zweiten Datei enthalten sind. Die dritte Spalte enthält schließlich die Zeilen, die in beiden Dateien enthalten sind.

Ein '-' anstelle eines Dateinamens steht für die Standardeingabe.

## Optionen:

`-{1,2,3}`

unterdrückt die erste, zweite bzw. dritte Spalte

## Siehe auch:

`diff`(info) [cmp](#), [sort](#) und [uniq](#)

## Autor:





Next: [cp](#) Up: [Von GNU's, Muscheln und](#) Previous: [comm](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# compress

## Funktion:

**compress** komprimiert Dateien

## Syntax:

```
compress [-cdfrvV] [-b Maxbits] [Datei ...]
```

## Beschreibung:

**compress** komprimiert Dateien mit LZW (einem veränderten Lempel-Ziv) Algorithmus. Bevor **compress** eine Datei durch die komprimierte Version ersetzt, überprüft es, ob die *Datei* nach der Kompression wirklich kleiner ist als vorher. Nur in diesem Fall wird die unkomprimierte *Datei* gelöscht. Wenn keine Datei angegeben wird, liest **compress** von der Standardeingabe und schreibt in die Standardausgabe. Wenn die Ausgabedatei *Datei.Z* schon existiert, wird sie nicht überschrieben. **compress** verändert den Zeitstempel der Datei nicht.

Die Programme **uncompress** und **zcat** sind Links auf **compress**, bei denen bestimmte Optionen vorbelegt sind.

**uncompress** arbeitet wie ``compress -d'`, das heißt, es entkomprimiert die mit **compress** gepackten Dateien.

**zcat** arbeitet wie ``compress -dc'`, das heißt, es schreibt die entkomprimierten Dateien auf die Standardausgabe. Diese Funktion wird in den Shellscripts **zdiff**, **zmore** oder **zless** benutzt, um den Inhalt komprimierter Dateien direkt an bestimmte Filter weiterzuleiten.

Das **compress** Programm kann als Standardpacker für Unix bezeichnet werden. Es wird im Zusammenhang mit **tar** auch für gepackte Dateiarchive verwendet. Das GNU **tar** bietet eine direkte Unterstützung von [compress](#).

Allerdings wird in letzter Zeit dazu übergegangen, das neue **gzip** Programm zum Packen einzelner Dateien zu benutzen, weil es deutlich höhere Kompressionsraten erzielt. Im Bereich der Freien

Software ist `compress` bereits überall durch `gzip` abgelöst.

## Optionen:

- c  
(compress) die (de-)komprimierte Datei wird in die Standardausgabe geschrieben
- d  
(decompress) dekomprimiert die Datei; die komprimierte Datei wird dabei ersetzt
- f  
(force) überschreibt existierende Ausgabedateien und ersetzt *Datei*, selbst wenn sie durch die Kompression nicht kleiner wird
- v  
(verbose) gibt den Dateinamen und das Größenverhältnis aus
- V  
(Version) gibt die Versionsnummer aus
- r  
(recursiv) komprimiert rekursiv alle Dateien in den Unterverzeichnissen
- b **Maxbits**  
setzt die Kompressionstiefe (Voreinstellung ist 16 Bit)

**Autoren:** Spencer W. Thomas, Jim McKie,  
Steve Davies, Ken Turkowski,  
James A. Woods, Joe Orost,  
Dave Mack und Peter Jannesen

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [cp](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [comm](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

**Next:** [cpio](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [compress](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Optionen:](#)
- 

# cp

## Funktion:

**cp** (copy) kopiert eine oder mehrere Dateien

## Syntax:

**cp** [*Optionen*] *Quelle* *Ziel*

**cp** [*Optionen*] *Quelle* ... *Verzeichnis*

## Optionen:

-a

(archiv) das gleiche wie -dpR

-b

(backup) sichert Dateien im Zielverzeichnis vor dem Überschreiben

-d

(no-dereference) kopiert die Links und nicht die Dateien, auf die der Link zeigt

-f

(force) Dateien im Zielverzeichnis werden überschrieben

-i

(interactive) erwartet Bestätigung vor dem Überschreiben bereits existierender Dateien

-l

(link) macht Links anstelle von Kopien (nur bei normalen Dateien)

-P

(path) die Quelldateien werden mit Pfad relativ zum Zielverzeichnis kopiert

-p

(preserve) erhält die Zugriffsrechte und Eigentümer des Originals (nicht die SUID und SGID Bits)

-r

(recursive) kopiert rekursiv alle Dateien der Unterverzeichnisse (auch Devices und Links) wie normale Dateien

-s

(symbolic link) macht symbolische Links anstelle von Kopien (absolute Pfadnamen)

-u

(update) überschreibt Ziel- nur durch neuere Quelldateien

-v

(verbose)

-x

(one file-system) ignoriert Unterverzeichnisse, die in anderen Dateisystemen angesiedelt sind

-R

(recursive) kopiert alle Unterverzeichnisse rekursiv; Spezialdateien bleiben erhalten

-S **Endung**

(suffix) sichert die Dateien vor dem Überschreiben durch Umbenennung mit der *Endung*; Voreinstellung ist '~'

-V {numbered, existing, simple}

(version-control) erhält auch frühere Versionen einer Datei, indem jeweils neue Backups erzeugt werden. Die Art der Backups kann auch durch die Umgebungsvariable `VERSION_CONTROL` bestimmt werden. Die Option `-V` überschattet `VERSION_CONTROL`. Wenn weder die Option noch die Environmentvariable gesetzt sind, wird `-existing` benutzt. Gültige Werte für `VERSION_CONTROL` sind:

numbered

Backups werden numeriert

existing

Backups werden nur für Dateien numeriert, die bereits numerierte Backups haben.

simple

es werden immer einfache Backups gemacht

## Autor:

Torbjorn Granlund, David MacKenzie und Jim Meyering

---

Next	Up	Previous	Contents	Index
------	----	----------	----------	-------

**Next:** [cpio](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [compress](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

Next: [csplit](#) Up: [Von GNU's, Muscheln und](#) Previous: [cp](#)

## Subsections

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)
- [Siehe auch:](#)

# cpio

## Funktion:

**cpio** erzeugt und verwaltet Dateiarhive verschiedener Formate

## Syntax:

```
cpio {-o|-create} [-0acvABLV] [-C Anzahl] [-H Format] [-M Nachricht]
[-O [[User@]Host:]Datei] [-F [[User@]Host:]Datei]
[-file=[[User@]Host:]Datei] [-format=Format] [-message=Nachricht]
[-null] [-reset-access-time] [-verbose] [-dot] [-append]
[-block-size=Größe] [-dereference] [-io-size=Größe] [-force-local]
[-help] [-version] < Liste [> Datei]
```

```
cpio {-i|-extract} [-bcdfmnrtsuvBSV] [-C Anzahl] [-E Datei] [-H
Format] [-M Nachricht] [-R [User][:.][Gruppe]] [-I
[[User@]Host:]Datei] [-F [[User@]Host:]Datei]
[-file=[[User@]Host:]Datei] [-make-directories] [-nonmatching]
[-preserve-modification-time] [-numeric-uid-gid] [-rename] [-list]
[-swap-bytes] [-swap] [-dot] [-unconditional] [-verbose]
[-block-size=Anzahl] [-swap-halfwords] [-io-size=Anzahl]
[-pattern-file=Datei] [-format=Format] [-owner=[user][:.][Gruppe]]
[-no-preserve-owner] [-message=Nachricht] [-force-local] [-help]
[-version] [Muster...] [< Datei]
```

```
cpio {-p|-pass-through} [-0adlmuvLV] [-R [user][:.][group]] [-null]
[-reset-access-time] [-make-directories] [-link]
[-preserve-modification-time] [-unconditional] [-verbose]
[-dereference] [-owner=[User][:.][Gruppe]] [-dot]
[-no-preserve-owner] [-help] [-version] Zielverzeichnis < Liste
```

## Beschreibung:

**cpio** ist ein Tool zur Erzeugung und Verwaltung von Dateiarchiven. In einem Dateiarchiv werden mehrere Dateien mit ihren Verzeichnissen und allen Verwaltungsinformationen, wie Eigentümer, Zugriffsrechte Erzeugungszeit etc., zu einer einzigen Datei oder zu einem Datenstrom zusammengefaßt. **cpio** erzeugt und verarbeitet eine ganze Reihe verschiedener Archivformate. Deshalb ist es besonders gut für den Austausch von Datenbeständen zwischen unterschiedlichen Rechnern geeignet.

**cpio** kann in drei verschiedenen Modi arbeiten.

Im copy-out Modus werden Daten aus dem Dateisystem in ein Archiv, zum Beispiel auf ein Magnetband, geschrieben. Die Namen der zu archivierenden Dateien liest **cpio** Zeilenweise von der Standardeingabe. Eine gängige Methode zur Erzeugung einer geeigneten Liste von Dateinamen ist die Verbindung des Ausgabekanals von [find](#) mit dem Eingabekanal von **cpio** durch eine Pipeline.

Im copy-in Modus werden die Daten vom Archiv in das Dateisystem kopiert. In diesem Modus liest **cpio** die archivierten Daten von der Standardeingabe. Wenn nicht das gesamte Archiv ausgepackt werden soll, können die gewünschten Dateien durch reguläre Ausdrücke nach den Optionen auf der Kommandozeile angegeben werden.

Im copy-pass Modus werden die Daten wie bei copy-out aus dem Dateisystem gelesen und sofort wieder in ein anderes Verzeichnis geschrieben, ohne daß zwischendurch ein Archiv erzeugt wird. Die Namen der zu kopierenden Dateien werden wie bei copy-out von der Standardeingabe gelesen, der Name des Zielverzeichnisses muß in der Kommandozeile nach den Optionen angegeben werden.

## Optionen:

- a  
veranlaßt **cpio**, nach dem copy-out die letzte Zugriffszeit vor dem Lesen zurückzusetzen
- A  
die Dateien werden an ein existierendes Archiv angehängt (nur im copy-out Modus auf Blockgeräten möglich)
- b  
veranlaßt **cpio**, beim extrahieren von Daten die Bytes von Datenwörtern und Halbwörtern zu tauschen
- B  
setzt die Blockgröße auf 5120 Bytes anstelle der voreingestellten 512 Bytes
- block-size=**Anzahl**  
setzt die Blockgröße auf *Anzahl*x512 Bytes
- c  
veranlaßt tar, das alte, portable ASCII Archivformat zu benutzen
- C **Größe**  
setzt die Blockgröße (*Größe* in Bytes)
- d  
veranlaßt tar, beim Auspacken eines Archivs die notwendigen Verzeichnisse zu erzeugen, wenn

sie noch nicht existieren

-E *Datei*

die Liste oder die regulären Ausdrücke zur Bestimmung der zu kopierenden Dateien wird aus der angegebenen Datei und nicht von der Standardeingabe gelesen

-f

verkehrt die Wirkung der Liste bzw. des Musters ins Gegenteil; es werden die Dateien kopiert, die nicht auf das Muster passen

-F **[[*User@* ]*Host:*]*Datei***

veranlaßt tar, die angegebene *Datei* als Archivdatei zu benutzen

-force-local

erzwingt die Interpretation eines Archivnamens bei den Optionen -F, -I und -O als lokale Datei, auch wenn in dem Dateinamen ein Doppelpunkt vorkommt

-H ***Format***

bestimmt eines der folgenden Archivformate (bei copy-in werden die unterstützten Formate automatisch erkannt):

bin

(Voreinstellung bei copy-out) veraltetes Binärformat

odc

das alte, portable POSIX-1 Format

newc

das neue, portable SVR4 Format; für große Dateisysteme mit mehr als 65536 I-Nodes geeignet

crc

wie newc mit zusätzlicher Prüfsumme

tar

das alte tar Format

ustar

das POSIX-1 tar Format und das GNU-tar Format

hpbm

das alte Binärformat des HPUX-cpio

hpodc

das portable POSIX-1 Format von HPUX; unterscheidet sich in der Speicherung von Gerätedateien

-i

schaltet cpio in den copy-in Modus; die in der Liste angegebenen Dateien werden aus dem Archiv in das System hinein kopiert

-I **[[*User@* ]*Host:*]*Datei***

verbindet die Standardeingabe von cpio mit der *Datei*; gegebenenfalls wird die Verbindung zum Rechner *Host* hergestellt und die Archivierung mit den Rechten von *User* ausgeführt

-k

ohne Funktion

-l

wenn möglich werden Dateien nicht kopiert sondern symbolische Links erzeugt

-L

im copy-out oder -pass Modus werden nicht die symbolischen Links kopiert, sondern die referenzierten Dateien

-m

das Datum der letzten Änderung bleibt beim Kopieren unverändert

-M **Nachricht**

veranlaßt cpio, die Nachricht auf die Standardfehlerausgabe zu schreiben, wenn das Backup-Medium voll ist; der Platzhalter '%d' kann benutzt werden, um in der Nachricht die laufende Nummer des aktuellen Bandes auszugeben (Start bei 1)

-n

die User- und Gruppen-ID der archivierten Dateien wird beim Listing in numerischer Form ausgegeben

-no-preserve-owner

(Voreinstellung für User ohne root-Privilegien) beim Extrahieren von Dateien aus dem Archiv oder beim Kopieren wird die archivierte User- und Gruppen-ID nicht auf die extrahierten Dateien übertragen

-O

schaltet cpio in den copy-out Modus; die in der Liste angegebenen Dateien werden aus dem System heraus kopiert und ein Archiv angelegt oder erweitert

-O [[*User*@ ]*Host*:]*Datei*

verbindet die Standardausgabe von cpio mit der *Datei*; gegebenenfalls wird die Verbindung zum Rechner *Host* hergestellt und die Archivierung mit den Rechten von *User* ausgeführt

-p

schaltet cpio in den copy-pass Modus

-r

erlaubt dem Anwender die interaktive Umbenennung von Dateien im copy-in Modus

-R [*User*][:.][*Gruppe* ]

die entsprechenden Benutzerrechte vorausgesetzt, werden Eigentümer und/oder Gruppe der Dateien beim Extrahieren geändert

-s

die Bytes eines Halbwortes werden beim Extrahieren der Daten vertauscht

-S

die Halbworte eines Wortes werden beim Extrahieren der Daten vertauscht

-t

zeigt den Inhalt des Archives an

-u

beim Extrahieren werden Dateien im Dateisystem ohne Nachfrage durch gleichnamige Dateien aus dem Archiv überschrieben, auch wenn diese älter sind

-v



zusammen mit -t wird ein ausführliches Listing des Archivinhalts ausgegeben

-V

für jede bearbeitete Datei wird ein Punkt in den Standardfehlerkanal geschrieben

-version

gibt die Versionsnummer von `cpio` aus

-0

die Elemente der Liste können durch Nullbytes anstelle der normalerweise erwarteten `NEWLINE` übergeben werden

## Siehe auch:

[tar](#), [ar\(1\)](#), [afio\(1\)](#)

## Autor:

Phil Nelson, David MacKenzie und John Oleynick

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [csplit](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [cp](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Beispiel:](#)
  - [Siehe auch:](#)
- 

# csplit

## Funktion:

**csplit** (context split) teilt eine Datei in mehrere Teile, wobei die Trennstelle durch ein Suchmuster angegeben werden kann

## Syntax:

```
csplit [-sk] [-f Prefix] [-n Stellen] [-prefix=Prefix]  
[-digits=Stellen] [-quiet] [-silent] [-keep-files] Datei Muster ...
```

## Beschreibung:

Mit **csplit** können Sie beliebige Textdateien an bestimmten kontextabhängigen Stellen zerteilen. Aus einer Eingabedatei (oder der Standardeingabe) werden mehrere Ausgabedateien erzeugt, denen Inhalt von einem *Suchmuster* abhängig gemacht werden kann. Die erste Zeile, in der das *Muster* vorkommt, wird zur ersten Zeile der nächsten Datei. Das *Muster* muß folgendermaßen angegeben werden:

*/Ausdruck/[Offset]* {*Anzahl*}

erzeugt eine Ausgabedatei, die alle Zeilen der Eingabe bis (ausschließlich) zu der Zeile mit dem *Ausdruck* enthält. Wird zusätzlich eine Zahl {*Anzahl*} in geschweiften Klammern angegeben, wird der Vorgang mit dem gleichen *Ausdruck* und dem verbleibenden Rest *Anzahl* mal wiederholt. Wird zusätzlich eine ganze Zahl mit einem führenden '+' oder '-' als *Offset* angegeben, so wird der Beginn der nächsten Datei um diese Anzahl Zeilen verschoben.

%*Ausdruck*%[*Offset*] {*Anzahl*}

arbeitet im Prinzip wie die vorher beschriebene Option, mit der Abweichung, daß keine Ausgabedatei erzeugt wird, dieser Teil der Eingabe also ignoriert wird.

*Nummer*

(eine einfache ganze Zahl) als Muster erzeugt eine Ausgabedatei aus den *Nummer* folgenden Zeilen.

Die Namen der Ausgabedateien bestehen aus dem *Prefix* und einer normalerweise zweistelligen Zahl. Der Standard für *Prefix* ist **xx**. Tritt während der Ausführung von `csplit` ein Fehler auf, so werden die bis dahin angelegten Dateien gelöscht.

## Optionen:

- s  
(silent) die Ausgabe der Ausgabedateigröße wird unterdrückt
- k  
(keep) bei einem Abbruch von `csplit` werden die bereits angelegten Dateien nicht gelöscht
- f *Prefix*  
die Ausgabedateien erhalten den *Prefix* als Namen
- n *Stellen*  
die Ausgabedateien erhalten Nummern mit der angegebenen Anzahl *Stellen*

## Beispiel:

Wenn Sie beispielsweise Ihre Mailbox (das ist die Datei, in der all ihre persönlichen eMails aneinandergehängt sind) auseinandernehmen wollen, können Sie das folgende Kommando versuchen:

```
$ csplit -k /var/spool/mail/$LOGNAME /^From / {100}
812
770
695
2279
2279
3201
975
1964
2946
csplit: `/^From /': match not found on repetition 9
1188
$ _
```

Ihre Mailbox mit ankommender Mail befindet sich standardmäßig im Verzeichnis `/var/spool/mail` und trägt Ihren Benutzernamen. Die einzelnen Mails werden durch sogenannte Header - also Briefköpfe - eingeleitet. Die Details zum Aufbau dieser Briefköpfe sind installationsabhängig. Die Suche nach einer Zeile, die mit `From` und einem darauffolgenden Leerzeichen beginnt, sollte aber mehr oder weniger exakt auf den Anfang des Mailheaders treffen.

Weil Sie wahrscheinlich nicht genau wissen, wie viele Mails in Ihrer Mailbox liegen, können Sie durch Angabe einer relativ hohen Zahl zusammen mit der `-k` Option erreichen, daß alle Mails getrennt und nach dem vorzeitigen Scheitern (hier nach dem 9. Durchgang) die bereits erzeugten Dateien nicht wieder gelöscht werden.

Die Dateien mit den einzelnen Mails werden im aktuellen Verzeichnis angelegt (Schreibberechtigung vorausgesetzt) und heißen `xx00` bis `xx09`.

## Siehe auch:

[split](#)

### Autor:

Stuart Kemp und David MacKenzie

---

Next	Up	Previous	Contents	Index
------	----	----------	----------	-------

**Next:** [cut](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [cpio](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Beispiel:](#)
- 

# cut

## Funktion:

**cut** schneidet bestimmte Teile aus den Zeilen einer Datei aus

## Syntax:

```
cut -b Bereich [-n] [Datei ...]
```

```
cut -c Bereich [Datei ...]
```

```
cut -f Bereich [-d Trenner] [-s] [Datei ...]
```

## Beschreibung:

**cut** liest aus den angegebenen Dateien oder von der Standardeingabe und gibt bestimmte Teile jeder Eingabezeile auf die Standardausgabe. Welcher Teil der Eingabezeile ausgegeben wird, hängt von der gewählten Option und der Wahl eines Bereiches ab. Ein *Bereich* ist eine durch Kommata getrennte Liste von einzelnen Zahlen oder Zahlenbereichen. Ein Zahlenbereich ist ein Ausdruck der Form ``m-n'`. Wird eine der Zahlen *m* oder *n* weggelassen, so wird der Zeilenanfang bzw. das Zeilenende angenommen.

## Optionen:

**-b *Bereich***

gibt nur die Bytes (Zeichen) im *Bereich* aus; TAB und BACKSPACE werden als ein Zeichen behandelt

**-c *Bereich***

gibt nur die Zeichen im *Bereich* aus; diese Option ist identisch mit der Option ``-b'`; TAB und BACKSPACE werden als ein Zeichen behandelt

**-f *Bereich***

gibt die Felder im *Bereich* aus; die einzelnen Felder sind durch `TAB` getrennt

`-d Trenner`

benutzt den *Trenner* anstelle eines `TAB` bei der Option ``-f'`

`-n`

ohne Funktion; vorgesehen für spätere Unterstützung internationaler Zeichensätze mit mehreren Bytes pro Zeichen

`-s`

unterdrückt die Ausgabe von Zeilen, die den *Trenner* nicht enthalten

## Beispiel:

Mit dem Kommando

```
$ cut -d : -f 1,5 /etc/passwd
ruth:Systemverwalterin
root:der traditionelle Superuser
daemon:der unbekannte Daemon
bin:
adm:
uucp:
news:Netnews Administrator
she:Sebastian Hetze
sync:
$ _
```

können Sie sich alle Benutzernamen (das 1. Feld) und die Realnamen (das 5. Feld) aller in `/etc/passwd` eingetragenen Accounts anzeigen lassen.

## Autor:

David M. Ihnat und David MacKenzie

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [date](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [csplit](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

**Next:** [dd](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [cut](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Die Systemzeit einstellen](#)
  - [Optionen:](#)
  - [Umgebung:](#)
  - [Beispiel:](#)
- 

# date

## Funktion:

**date** schreibt oder setzt die Systemzeit

## Syntax:

```
date [-u] [-s Datum] [+Formatstring] [MMDDhhmm[[CC]YY][.ss]]
```

## Beschreibung:

Das **date**-Kommando liest oder setzt die Linux Systemzeit.

Weniger zur Datumsausgabe auf der Kommandozeile als vielmehr zur Benutzung in Shellscripts kann **date** das Datum in vielfältiger Weise formatiert ausgeben. Es arbeitet dabei im Prinzip als Frontend zur `strftime(3)` C-Bibliotheksfunktion. Ein durch ``+'` eingeleiteter Formatstring wird durch entsprechende Datumsangaben ergänzt und auf dem Standardausgabekanal angezeigt.

In einem vollständig eingerichteten Linux System paßt sich **date** automatisch an die lokale Zeitzone an. Insbesondere findet der Wechsel zwischen Mitteleuropäischer Zeit (MET) und Sommerzeit (Daylight Saving Time (DST), MEST) automatisch statt. Die Zeitzone kann mit der Umgebungsvariablen `TZ` verändert werden.

Durch Belegung einer der Umgebungsvariablen `LANG` oder `LC_TIME` mit einem gültigen Wert ([siehe Locales](#)) kann das Ausgabeformat von **date** auch im länderspezifischen Format erfolgen.

In dem durch ein ``+'` eingeleiteten Formatstring werden die folgenden Parameter unterstützt:

`%%`

ein einfaches `%`

**%n**  
das Zeilenende

**%t**  
ein Tabulator

## Die Zeitfelder:

**%H**  
Stunden, 00 bis 23

**%I**  
Stunden, 00 bis 12

**%k**  
Stunden, 0 bis 23

**%l**  
Stunden, 1 bis 12

**%M**  
Minuten 00 bis 59

**%p**  
AM oder PM bzw. die lokale Bezeichnung für Vor- und Nachmittag

**%r**  
die Zeit, 12 Stunden (hh:mm:ss AM/PM)

**%R**  
die Zeit, hh:mm

**%s**  
Sekunden seit dem 1.1.1970, 00:00:00 Uhr (Epoche)

**%S**  
Sekunden 00 bis 59

**%T**  
die Zeit, 24 Stunden (hh:mm:ss)

**%X**  
die Zeit, 24 Stunden (%H:%M:%S)

**%Z**  
die Zeitzone; oder nichts, wenn keine Zeitzone feststellbar ist

## Die Datenfelder:

**%a**  
der abgekürzte Wochentag im lokalen Format

**%A**  
der ausgeschriebene Wochentag im lokalen Format

**%b**  
der abgekürzte Monat im lokalen Format



%B	der ausgeschriebene Monat im lokalen Format
%c	das Datum und die Zeit im bevorzugten lokalen Format
%C	das Jahrhundert
%d	der Tag im Monat, 01 bis 31
%D	das Datum (mm/dd/yy)
%e	der Tag im Monat, 1 bis 31
%h	das gleiche wie %b
%j	der Tag im Jahr
%m	der Monat 00 bis 12
%U	die Nummer der Woche, Sonntag erster Tag
%w	der Tag in der Woche 0 bis 6
%W	die Nummer der Woche, Montag erster Tag
%x	das Datum (mm/dd/yy)
%y	die letzten beiden Stellen der Jahreszahl
%Y	die ganze Jahreszahl

## Die Systemzeit einstellen

Die Ein- bzw. Umstellung der Systemzeit ist nur der Superuserin erlaubt.

Die Angabe des neuen Datums kann entweder mit der `-s` Option im Format der normalen Datumsausgabe erfolgen, oder sie erfolgt in Form einer einzigen Zahl. Die einzelnen Stellen der Zahl haben folgende Bedeutung:

MM  
Monat

DD

	Tag im Monat
hh	
	Stunde
mm	
	Minute
CC	
	die ersten beiden Stellen der Jahreszahl (optional)
YY	
	die letzten beiden Stellen der Jahreszahl (optional)
ss	
	die Sekunden (optional)

## Optionen:

- d *Datum*  
wandelt das angegebene *Datum* in das Standardformat oder ein anderes in der Kommandozeile definiertes Format um
- s *Datum*  
(set) setzt die Zeit auf das *Datum*; das *Datum* kann Monatsnamen, Zeitzonen und ähnliches enthalten
- u  
(universal) zeigt (oder setzt) die Zeit als UCT (Universal Coordinated Time, auch bekannt als Greenwich Mean Time)

## Umgebung:

In der TZ Umgebungsvariablen kann eine andere als die von der Systemverwalterin vorgegebene Zeitzone eingestellt werden. Wenn keine nach der Zeitzone benannte Datei in `/usr/lib/zoneinfo` existiert, wird automatisch die GMT benutzt.

Wenn die aktuelle Version von `date` Locales unterstützt kann in der Umgebungsvariablen `LANG` oder `LC_TIME` ein lokales Datumsformat bestimmt werden. Der Wert dieser Variablen muß auf eines der Unterverzeichnisse von `/usr/lib/locale` passen und in diesem Verzeichnis muß eine Datei `LC_TIME` mit der gewünschten `tminfo`-Struktur existieren. Zusätzliche Informationen zu Locales finden Sie [hier](#).

## Beispiel:

Mit dem Kommando

```
$ date "+Es ist %H Stunden und %M Minuten nach Mitternacht."
Es ist 16 Stunden und 03 Minuten nach Mitternacht.
$ _
```

können Sie auf einfache Art eine umständliche Uhrzeit anzeigen lassen.

Wenn Sie Superuserrechte haben, können Sie die Systemzeit einstellen:

```
# date -s "Thu Sep 16 15:09:07 GMT 1993"
Thu Sep 16 17:09:07 MET DST 1993
# _
```

Das Datum wird intern durch eine vorzeichenbehaftete vier Byte Integer Variable repräsentiert. Der „Nullpunkt“ ist am 1.1.1970 um 0 Uhr. Dieses Datum wird auch als Epoche bezeichnet. Der Bereich zulässiger Systemzeiten beginnt am 13.12.1901 um 20:45:52 und endet am 19.01.2038 um 03:14:07 Universal Coordinated Time.

## Autor:

David MacKenzie

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [dd](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [cut](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

Next: [df](#) Up: [Von GNU's, Muscheln und](#) Previous: [date](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Beispiel:](#)
- 

# dd

## Funktion:

**dd** (disk dump) konvertiert Dateien für verschiedene Speichermedien

## Syntax:

```
dd [if=Datei] [of=Datei] [ibs=Bytes] [obs=Bytes] [bs=Bytes]
[cbs=Bytes] [skip=Blöcke] [seek=Blöcke][count=Blöcke] [conv={ascii,
ebcdic, ibm, block, unblock, lcase, ucase, swab, noerror, notrunc,
sync}]
```

## Beschreibung:

**dd** liest eine Datei und schreibt den Inhalt mit wählbarer Blockgröße und verschiedenen Konvertierungen. Mit Hilfe dieses Kommandos können reguläre Dateien ebenso wie ganze Disketten oder Festplattenpartitionen kopiert werden.

## Optionen:

*if=Datei*

(input file) der Name der Eingabedatei (voreingestellt ist die Standardeingabe)

*of=Datei*

(output file) der Name der Ausgabedatei (voreingestellt ist die Standardausgabe)

*ibs=Schritt*

(input block size) Blockgröße der Eingabedatei

*obs=Schritt*

(output block size) Blockgröße der Ausgabedatei

*bs=Schritt*

(block size) Blockgröße für Ein- und Ausgabedatei

cbs=**Schritt**

(conversion block size) Blockgröße für Konvertierung

skip=**Blocks**

ignoriert am Anfang die angegebene Anzahl *Blocks* von der Eingabe

seek=**Blocks**

unterdrückt am Anfang die Ausgabe der angegebenen Anzahl *Blocks*

count=**Blocks**

kopiert die angegebene Anzahl *Blocks*

conv=**Konvertierung** ...

bestimmt die Art der Konvertierung; *Konvertierung* ist dabei eine von:

ascii

konvertiert EBCDIC nach ASCII

ebcdic

konvertiert ASCII nach EBCDIC

ibm

konvertiert ASCII nach big blue special EBCDIC

block

schreibt Zeilen in Felder der Größe *cbs* und ersetzt das Zeilenende durch Leerzeichen; der Rest des Feldes wird ebenfalls mit Leerzeichen aufgefüllt

unblock

ersetzt abschließende Leerzeichen eines Blocks der Größe *-cbs* durch ein Zeilenende

lcase

wandelt Großbuchstaben in Kleinbuchstaben

ucase

wandelt Kleinbuchstaben in Großbuchstaben

swab

vertauscht je zwei Bytes der Eingabe; wenn die Anzahl der gelesenen Bytes ungerade ist, wird das letzte Byte einfach kopiert

noerror

ignoriert Lesefehler

sync

füllt Eingabeblocks bis zur Größe von *ibs* mit Nullen

## Beispiel:

Das Kommando

```
$ dd bs=8192 if=zImage of=/dev/fd0  
26+1 records in  
26+1 records out
```

\$ \_

können Sie benutzen, um die fertig übersetzte Kerneldatei (zImage) auf eine formatierte Diskette zu schreiben und so eine Bootdiskette zu erzeugen.

Mit dem Kommando

```
# dd if=/dev/hda of=/dev/fd0 bs=512 count=1
1+0 records in
1+0 records out
# _
```

kann die Superuserin (Ruth) eine Kopie des Festplattenbootsektors auf einer Diskette anlegen. Mit dieser Diskette kann die Festplatte gebootet werden, wenn der Festplattenbootsektor zerstört wurde.

### **Autor:**

Paul Rubin, David MacKenzie und Stuart Kemp

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [df](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [date](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

**Next:** [dirname](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [dd](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# df

## Funktion:

**df** (disk free) zeigt den freien Festplattenplatz

## Syntax:

```
df [-aikPv] [-t Fstyp] [-all] [-inodes] [-type fstype] [-kilobytes]  
[-portability] [Pfad ...]
```

## Beschreibung:

**df** zeigt den freien Festplattenplatz für das Dateisystem, in dem das Verzeichnis *Pfad* angesiedelt ist. Wenn kein Verzeichnis angegeben ist, wird der freie Platz für alle aufgesetzten Dateisysteme angezeigt. Die Angabe erfolgt in Kilobyte, wenn nicht die Umgebungsvariable POSIXLY\_CORRECT gesetzt ist. In diesem Fall wird der freie Platz in 512-Byte Sektoren angezeigt.

Wird als *Pfad* der absolute Name der Gerätedatei eines aufgesetzten Dateisystems angegeben, so wird der freie Platz auf diesem Gerät (Partition einer Festplatte) angegeben und nicht der des Dateisystems, in dem sich die Gerätedatei befindet. Der freie Platz auf einem abgesetzten Dateisystem ist auf diese Weise nicht zu ermitteln.

## Optionen:

-a

(all) zeigt alle Dateisysteme an, einschließlich der mit 0 Bytes Kapazität und der vom Typ ignore oder auto

-i

(inodes) zeigt die Auslastung der Inodes anstelle der Blockauslastung

-k

(kilobytes) zeigt den freien Platz in Kilobyteblöcken, auch wenn die Umgebungsvariable POSIXLY\_CORRECT gesetzt ist

-P

(portability) erzwingt die Ausgabe in einer Zeile pro Dateisystem

-t *Fstyp*

(type) schränkt die Ausgabe auf die Dateisysteme vom Typ *Fstyp* ein

## Autor:

David MacKenzie

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [dirname](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [dd](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)



**Next:** [doshell](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [df](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Siehe auch:](#)
- 

# dirname

## Funktion:

**dirname** gibt den Pfadanteil eines vollständigen Dateinamens aus

## Syntax:

**dirname** *Datei*

## Beschreibung:

**dirname** schneidet aus einem Dateinamen mit absoluter Pfadangabe den eigentlichen Dateinamen ab und liefert den bloßen Verzeichnisnamen.

## Siehe auch:

[basename](#) und bei der [bash](#) und [bash](#)

## Autor:

David MacKenzie und Jim Meyering

**Next:** [du](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [dirname](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
- 

# doshell

## Funktion:

`doshell` startet eine Shell auf einem (virtuellen) Terminal ohne `login`

## Syntax:

`doshell` *Terminal Shell*

## Beschreibung:

`doshell` ermöglicht es, auch auf den (virtuellen) Terminals, auf denen von `init` keine `getty`-Prozesse gestartet wurden und auf denen deshalb kein `login` möglich ist, eine Shell zu starten. Die Umschaltung zwischen den virtuellen Terminals erfolgt mit den Tastenkombinationen `ALT-F1` bis `ALT-F8`.

## Autor:

Jim Wiegand

Next: [echo](#) Up: [Von GNU's, Muscheln und](#) Previous: [doshell](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# du

## Funktion:

**du** (disk usage) zeigt die Verteilung des belegten Plattenplatzes auf die Verzeichnisse

## Syntax:

```
du [-abcklsxDLS] [-all] [-total] [-count-links] [-summarize] [-bytes]
[-kilobytes] [-one-file-system] [-separate-dirs] [-dereference]
[-dereference-args] [Verzeichnis ...]
```

## Beschreibung:

**du** zeigt den belegten Plattenplatz für das *Verzeichnis* und für alle Unterverzeichnisse (in Kilobyte). Wenn die Umgebungsvariable `POSIXLY_CORRECT` gesetzt ist, wird die Menge in 512 Byte Blöcken angegeben.

## Optionen:

- a  
(all) zeigt auch den Platzbedarf aller Dateien
- b  
(bytes) zeigt den Platzbedarf in Bytes
- c  
zeigt den (summierten) Platzbedarf der in der Kommandozeile übergebenen Dateien
- k  
(kilobytes) gibt den Platzbedarf in Kilobytes, auch wenn die Umgebungsvariable `POSIXLY_CORRECT` gesetzt ist
- l  
zählt die Größe der (harten) Links mit, auch wenn sie dadurch doppelt vorkommen

-S

gibt nur die Summe für jedes Verzeichnis in der Kommandozeile

-x

ignoriert Verzeichnisse, die in anderen Dateisystemen liegen

-D

folgt dem Verweis auf ein anderes Verzeichnis bei einem symbolischen Link, wenn dieser als Kommandozeilenargument übergeben wird. Andere symbolische Links werden nicht dereferenziert.

-L

alle symbolischen Links werden dereferenziert, das heißt es wird der Platzbedarf des referenzierten Verzeichnisses anstelle des Linkfiles gezeigt

-S

zeigt den Platzbedarf jedes Verzeichnisses einzeln, ohne die Unterverzeichnisse

## Autor:

Torbjorn Granlund und David MacKenzie

---

Next	Up	Previous	Contents	Index
------	----	----------	----------	-------

**Next:** [echo](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [doshell](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

**Next:** [egrep](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [du](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- 

# echo

## Funktion:

**echo** gibt eine oder mehr Zeichenketten auf die Standardausgabe

## Syntax:

**echo** [-ne] *Zeichenkette* ...

## Beschreibung:

**echo** gibt eine oder mehr Zeichenketten auf die Standardausgabe. Die einzelnen Zeichenketten werden durch Leerzeichen voneinander getrennt, und die Letzte wird durch einen Zeilenvorschub abgeschlossen.

## Optionen:

-n

unterdrückt den abschließenden Zeilenvorschub

-e

ermöglicht die folgenden Sonderzeichen in der Zeichenkette:

\a

Alarmton

\b

Rückschritt

\c

unterdrückt den abschließenden Zeilenvorschub

\f

	Seitenvorschub
<code>\n</code>	
	Zeilenvorschub
<code>\r</code>	
	Wagenrücklauf
<code>\t</code>	
	Tabulator
<code>\v</code>	
	vertikaler Tabulator
<code>\\</code>	
	Backslash
<code>\nnn</code>	
	das Zeichen mit der (oktalen) Nummer <i>nnn</i>

## Siehe auch:

Das `echo`-Kommando ist auch ein eingebautes Shellkommando in der Bash. Die Beschreibung [hier](#) zu finden.

## Autor:

Brian Fox und Chet Ramey

**Next:** [elvis](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [echo](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Beispiel:](#)
  - [Siehe auch:](#)
- 

# egrep

## Funktion:

**egrep** gibt alle Zeilen aus, in denen ein bestimmter Ausdruck gefunden wird

## Syntax:

**egrep** [-CVbchilnsvwX ] [-Anzahl] [-AB Anzahl] [[-e ] *Ausdruck* | -f *Datei*] [*Datei* ...]

## Beschreibung:

**egrep** durchsucht die angegebenen *Dateien* (oder die Standardeingabe) nach einem *Ausdruck* und gibt die entsprechenden Zeilen aus. Der Status von **egrep** ist 0, wenn der *Ausdruck* gefunden wurde, und sonst 1.

**egrep** unterscheidet sich nur in der Syntax einiger Ausdrücke vom **grep**-Kommando. Als *Ausdruck* akzeptiert **egrep** reguläre Ausdrücke mit den folgenden Steuerzeichen: 

**c**

ein einzelner Buchstabe paßt auf sich selbst

**.**

ein Punkt paßt auf jeden Buchstaben, außer auf das Zeilenende

**?**

das dem Fragezeichen vorangehende Zeichen bzw. Muster kann null oder einmal vorkommen

**\***

das dem Asterisk '\*' vorangehende Zeichen oder Muster kann 0 mal oder öfter vorkommen

**+**

das dem Pluszeichen '+' vorangehende Zeichen bzw. Muster kann 1 mal oder öfter vorkommen

die durch den Operator '|' verbundenen Argumente werden **oder** verknüpft

(Caret) paßt auf den Zeilenanfang

paßt auf das Zeilenende

paßt auf den Wortanfang

paßt auf das Wortende

### [**Buchstaben**]

paßt auf alle *Buchstaben*; dabei können einzelne Buchstaben, aber auch Bereiche in der Form '*von-bis*' angegeben werden; wenn der erste Buchstabe nach '[' ein '^' ist, paßt der Ausdruck auf alle Buchstaben außer den aufgeführten

die Klammern fassen Ausdrücke und Zeichenketten zusammen; außerdem wird der auf den in Klammern eingeschlossene Teil des Musters passende Text markiert und mit einem folgenden '\N' Ausdruck referenziert (Tag)

referenziert die auf das in den N-ten runden Klammern eingeschlossene Muster passende Zeichenkette.

jedes der Sonderzeichen kann, durch ein '\'(Backslash) eingeleitet, sich selbst suchen

paßt auf kein Zeichen, sondern auf den Anfang oder das Ende eines Wortes

steht für den Raum innerhalb eines Wortes

paßt auf alle alphanumerischen Zeichen [A-Za-z0-9]

paßt auf alle nicht alphanumerischen Zeichen [^A-Za-z0-9]

Die Rangfolge der Operatoren ist (von der höchsten zur niedrigsten):

'(', ')', '?', '\*', '+' und '|'

Die anderen Operatoren sind mit den anderen Buchstaben gleichrangig.



# Optionen:

- A **Anzahl**  
gibt *Anzahl* Zeilen Kontext **nach** jeder gefundenen Zeile aus
- B **Anzahl**  
gibt *Anzahl* Zeilen Kontext **vor** jeder gefundenen Zeile aus
- C  
gibt 2 Zeilen Kontext **vor und nach** jeder gefundenen Zeile aus
- Anzahl**  
gibt *Anzahl* Zeilen Kontext **vor und nach** jeder gefundenen Zeile aus
- V  
gibt die Versionsnummer auf die Standardfehlerausgabe
- b  
gibt die Position jeder gefundenen Stelle mit aus
- c  
gibt nur die Gesamtzahl der gefundenen Stellen aus
- e **Ausdruck**  
sucht nach *Ausdruck*
- f **Datei**  
*Datei* enthält die Ausdrücke, nach denen gesucht werden soll
- h  
unterdrückt die Dateinamen vor jeder Fundstelle
- i  
ignoriert Groß- und Kleinschreibung
- l  
gibt nur die Dateinamen mit Fundstellen aus
- n  
gibt die Zeilennummer zu jeder Fundstelle aus
- s  
(silent) keine Ausgabe außer Fehlermeldungen
- v  
gibt nur Zeilen aus, die den *Ausdruck* nicht enthalten
- w  
gibt nur Zeilen aus, in denen der *Ausdruck* als komplettes Wort vorkommt
- x  
gibt nur Zeilen aus, die den *Ausdruck* als ganze Zeile enthalten

# Beispiel:

## Das Kommando

```
$ egrep -h 'sys_.*[^;]$\' *.c
int sys_waitpid(pid_t pid,unsigned long * stat_addr, int options)
int sys_fork(struct pt_regs regs)
int sys_sysinfo(struct sysinfo *info)
int sys_ioperm(unsigned long from, unsigned long num, int turn_on)
int sys_syslog(int type, char * buf, int len)
int sys_ptrace(long request, long pid, long addr, long data)
int sys_pause(void)
int sys_alarm(long seconds)
$ _
```

liefert, im Verzeichnis `/usr/src/linux/kernel` ausgeführt, die erste Näherung einer Liste aller Systemaufrufe von Linux. Es werden alle C-Sourcen nach Zeilen durchsucht, die die für die Funktionsnamen der Systemaufrufe typischen Buchstaben `sys_` enthalten und die gleichzeitig nicht mit einem Semikolon enden.

Die beiden in diesem Beispiel verwendeten Asterisk Wildcards ``*'`` haben grundlegend unterschiedliche Funktion: das erste, in Hochkommata eingeschlossene wird dem `egrep`-Kommando übergeben und mit dem Ausdruck ausgewertet. Das zweite wird von der Shell interpretiert, die daraus eine Liste aller Dateinamen mit der Endung ``.c'` im aktuellen Verzeichnis erzeugt.

## Mit dem Kommando

```
$ egrep '(wird).*\1' Handbuch.tex
erzeugt wird, dieser Teil der Eingabe also ingoriert wird.
nicht existiert, wird sie erzeugt, sonst wird sie erweitert; die
bearbeitet wird, wird ein Prozeß manchmal erst mit dem Signal
gestartet wird, kein Eintrag gefunden wird, gibt \kommando{logname}
Datei verzweigt, so wird sie überschrieben, anderenfalls wird sie
$ _
```

kann in der Datei `Handbuch.tex` nach Zeilen gesucht werden, in denen das Wort `wird` doppelt vorkommt.

# Siehe auch:

[grep](#)

## Autor:

Mike Haertel, James A. Woods und David Olson

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [elvis](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [echo](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

## Subsections

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)
- [Überblick](#)
- [Der `visual mode'](#)
  - [`Visual mode' Befehle](#)
  - [Kommandos zum Positionieren des Cursors:](#)
  - [Kommandos zum Ändern von Text:](#)
  - [Arbeiten mit Puffern und Marken:](#)
  - [Kommandos zum Suchen nach Ausdrücken und Wörtern:](#)
  - [Wiederholungs Kommandos:](#)
  - [Sonstige Kommandos:](#)
  - [Eingabemodi](#)
  - [Die Cursortasten in den Eingabemodi](#)
  - [Digraphs](#)
  - [Abkürzungen](#)
  - [Automatisches Einrücken](#)
- [Der `colon mode'](#)
  - [`Colon mode' Befehle](#)
  - [Zeilenangaben](#)
  - [Texteingabe Kommandos](#)
  - [Ausschneiden und Einfügen](#)
  - [Kommandos zum Anzeigen von Text](#)
  - [Kommandos, die auf den gesamten Text wirken](#)
  - [Kommandos zum Editieren einzelner Zeilen](#)
  - [Der undo Befehl](#)
  - [Konfiguration und Status](#)
  - [Kommandos zum Arbeiten mit mehreren Dateien](#)
  - [Zwischen Dateien wechseln](#)
  - [Arbeiten mit einem Compiler](#)
  - [elvis beenden](#)

- [Datei Ein- und Ausgabekommandos](#)
  - [Verzeichnis Kommandos](#)
  - [Debugging Kommandos](#)
  - [Reguläre Ausdrücke](#)
    - [Funktion](#)
    - [Ersetzungen](#)
    - [Optionen](#)
    - [Beispiele](#)
  - [Die Optionen von elvis](#)
  - [Zwischenspeicher \(Puffer\)](#)
    - [Text zwischenspeichern](#)
    - [Einfügen aus einem Puffer](#)
    - [Makros](#)
    - [Wechseln der Arbeitsdatei](#)
- 

# elvis

## Funktion:

**elvis** ist eine Weiterentwicklung, des Standard Unix Editors ex/vi.

## Syntax:

**elvis** [-reviR] [-t *tag*] [-m [*Datei*]] [-w *Fenstergröße*] [-c *Befehl*] [+*Befehl*]

## Beschreibung:

**elvis** bietet sowohl im `visual mode', als auch im `colon mode' nahezu alle Möglichkeiten des `Vorbilds' ex/vi.

**elvis** kann unter verschiedenen Namen aufgerufen werden. Wird **elvis** als „vi“ aufgerufen, so verhält er sich wie **elvis**. Unter dem Namen „view“ läßt **elvis** keine Änderungen an der Datei zu und schützt sie so vor versehentlichem Überschreiben. Als „ex“ startet **elvis** im „colon mode“ (wie -e, und als „input“ aufgerufen, befindet man sich sofort im „input mode“, als sei die Option „-i“ gesetzt.

Wie ex/vi hält auch **elvis** den Hauptteil des Textes in einer temporären Datei und nicht im RAM. Dadurch können auch Dateien editiert werden, die wegen ihrer Größe nicht im Speicher gehalten werden können. Im Falle eines Systemabsturzes besteht so die Möglichkeit, den Inhalt des Arbeitspuffers durch das Hilfsprogramm **elvrec** wiederherzustellen.

elvis ist ein Public-Domain Programm und unterliegt keinen Beschränkungen in der Verwendung.

## Optionen:

-r

Bei dem originalen ex/vi bedeutet die „-r“ Option, daß ein durch einen Absturz oder Stromausfall unterbrochener Editiervorgang wiederhergestellt werden soll. elvis verwendet hierfür ein separates Programm namens *elvrec* und gibt deshalb nur einen Hinweis auf dieses Programm aus.

-R

Diese Option setzt das *readonly* Flag, so daß eine Datei nicht versehentlich überschrieben werden kann.

-t *Tagname*

elvis positioniert den Cursor nach dem Starten auf den angegebenen Referenzpunkt *Tagname*.

-m [*Datei*]

elvis durchsucht die angegebene Datei nach Compilerfehlermeldungen, und plaziert den Cursor in der Datei auf der Zeile, in der der Fehler aufgetreten ist. Wird keine Datei angegeben, so sucht elvis in der Datei „errlist“.

-e

elvis startet im „colon mode“.

-v

elvis startet im „visual command mode“.

-i

elvis startet im „input mode“.

-w *Fenstergröße*

Setzt die „window“ Option auf den Wert von *Fenstergröße*.

-c *Befehl* oder +*Befehl*

Nachdem elvis die erste Datei geladen hat, führt elvis *Befehl* als „ex“ Kommando aus. Ein typisches Beispiel hierfür ist „*elvis +75 Datei*“, um den Cursor automatisch auf Zeile 75 der *Datei* zu plazieren.

## Überblick

Die Benutzerschnittstelle von elvis ist etwas gewöhnungsbedürftig. Es gibt zwei wichtige Kommandomodi und einige Eingabemodi. Jeder Kommandomodus hat einen Befehl, um in den jeweils anderen umzuschalten. Der meistverwendete Modus ist vermutlich der 'visual command mode'. In diesem Modus befindet sich elvis normalerweise nach dem Starten.

Im 'visual command mode' dient der gesamte Bildschirm der Textdarstellung. Jeder Tastendruck wird als Teil eines Befehls interpretiert. Eingegebener Text wird **nicht** in die Datei eingefügt. Um Text einzufügen, muß erst ein 'Text einfügen' Kommando gegeben werden.

Der 'colon mode' unterscheidet sich vom 'visual mode' dadurch, daß in der letzten Zeile ein ':'

angezeigt wird. `elvis` erwartet dann die Eingabe eines Befehls, gefolgt von einem `RETURN`. Im ``colon mode'` erwartet `elvis` andere Kommandos als im ``visual command mode'`.

# Der ``visual mode'`

## ``Visual mode' Befehle`

Die meisten ``visual command mode'` Befehle sind nur einen Tastendruck lang. Nachfolgende Tabelle listet die möglichen Befehle und ihre Optionen auf.

Zusätzlich zu den hier aufgeführten Kommandos interpretiert `elvis` die Cursortasten als entsprechende Positionierungsbefehle, sofern der entsprechende *termcap*-Eintrag korrekt ist. Hier treten gelegentlich Probleme auf. Gleiches gilt für die `BILDAUF`- und `BILDAB`-Tasten. Im ``colon mode'` stellt `elvis` noch einen Befehl (`:map`) zur Verfügung, der es ermöglicht noch andere Tasten, wie z. B. die Funktionstasten, mit Kommandos zu belegen.

Noch ein Tip: der ``visual command mode'` ist dem ``text input mode'` sehr ähnlich. Um herauszufinden in welchem Modus man sich gerade befindet, genügt es, einmal `ESC` zu drücken. Befindet man sich im ``visual command mode'`, piepst `elvis` einmal. Piepst `elvis` nicht, so hat man sich im ``input mode'` befunden, und `elvis` ist in den ``visual command mode'` zurückgekehrt, d. h. nach einem `ESC` befindet `elvis` sich immer im ``visual command mode'`.

### **Anzahl**

Vielen Kommandos kann ein numerischer Parameter vorangestellt werden. Er besteht aus einer Folge von Ziffern, die als Dezimalzahl interpretiert werden. Dieser Parameter ist immer optional. Sein Standardwert ist meistens 1.

### **Taste**

Einige Kommandos erfordern zwei Tastendrucke. Der erste ist immer der Befehl, der zweite ein Parameter zum Befehl. So bedeutet *ma* beispielsweise, daß an die aktuelle Cursorposition die Marke *a* gesetzt wird.

### **Bereich**

Kommandos, die auf einen Textbereich wirken, benötigen zusätzlich zur aktuellen Cursorposition, die eine Grenze eines Bereichs angibt, eine zweite. `elvis` kennt drei Möglichkeiten, diese zweite Grenze anzugeben. Die erste ist, dem Befehl ein Bewegungskommando nachzustellen. So löscht *dw* ein Wort, und sowohl *3dw* als auch *d3w* löschen jeweils drei Wörter. Als zweite Möglichkeit kann man den Befehl zweimal angeben. Er wirkt dann auf die ganze Zeile. Die dritte ist, den Cursor auf ein Ende des Bereichs zu setzen, *v* oder *V* einzugeben, den Cursor an das andere Ende des Bereichs zu bewegen und daraufhin den gewünschten Befehl einzugeben.

### **eing**

Bei einigen Kommandos ist es möglich, interaktiv Text einzugeben. Es erscheint entweder ein Prompt, oder `elvis` schaltet in den sog. ``input mode'` (s. u.).

### **Text**

Bei Kommandos, die einen Parameter *Text* erwarten, kann der gewünschte Text in der letzten Bildschirmzeile interaktiv eingegeben werden. Die Texteingabe wird mit `RETURN` abgeschlossen.

### **BEW**

Diese Kommandos bewegen den Cursor, und können einem Befehl übergeben werden, der

einen *Bereich*-Parameter erwartet.

## EDIT

Diese Kommandos modifizieren Text und können mit `.' wiederholt werden.

## EXT

Diese Befehle sind Erweiterungen von `elvis`, die bei `ex/vi' nicht zur Verfügung stehen.

## Kommandos zum Positionieren des Cursors:

### Befehl    Beschreibung

- Anzahl* & *h* & & Cursor rechts *Anzahl* Spalten (BEW)
- Anzahl* & ^*H* & & Cursor rechts (wie *h*) *Anzahl* Spalten (BEW)
- Anzahl* & *l* & & Cursor links *Anzahl* Spalten (BEW)
- Anzahl* & `SPC` & & Cursor links *Anzahl* Spalten (wie *l*) (BEW)
- Anzahl* & ^*X* & & Cursor zur Spalte *Anzahl* (BEW) (ERW)
- Anzahl* & | & & Cursor zur Spalte *Anzahl* (BEW)
- Anzahl* & *t* & *Taste* & Cursor nach rechts **vor** das Zeichen *Taste* (BEW)
- Anzahl* & *f* & *Taste* & Cursor nach rechts **auf** das Zeichen *Taste* (BEW)
- Anzahl* & *T* & *Taste* & Cursor nach links **vor** das Zeichen *Taste* (BEW)
- Anzahl* & *F* & *Taste* & Cursor nach links **auf** das Zeichen *Taste* (BEW)
- Anzahl* & *j* & & Cursor abwärts *Anzahl* Zeilen (BEW)
- Anzahl* & ^*J* & & Cursor abwärts (wie *j*) *Anzahl* Zeilen (BEW)
- Anzahl* & ^*N* & & Cursor abwärts *Anzahl* Zeilen (BEW)
- Anzahl* & *k* & & Cursor aufwärts *Anzahl* Zeilen (BEW)
- Anzahl* & ^*P* & & Cursor aufwärts *Anzahl* Zeilen (BEW)
- Anzahl* & *w* & & Cursor rechts *Anzahl* Wörter (BEW)
- Anzahl* & *W* & & Cursor rechts *Anzahl* Wörter (BEW)
- Anzahl* & *b* & & Cursor links *Anzahl* Wörter (BEW)
- Anzahl* & *B* & & Cursor links *Anzahl* Wörter (BEW)
- Anzahl* & *e* & & Cursor zum Wortende *Anzahl* Wörter (BEW)
- Anzahl* & *E* & & Cursor zum Wortende *Anzahl* Wörter (BEW)
- & ^ & & Cursor zum Zeilenanfang (BEW)
- & 0 & & Cursor zum Zeilenanfang, wenn nicht Teil einer Wiederholungsangabe

*Anzahl* & \_ & & Cursor zum Zeilenanfang oder *Anzahl* Zeilen abwärts (BEW)

& \$ & & Cursor zum Zeilenende (BEW)

*Anzahl* & ^M & & Cursor abwärts zum Zeilenanfang *Anzahl* Zeilen (BEW)

*Anzahl* & + & & Cursor abwärts zum Zeilenanfang *Anzahl* Zeilen (BEW)

*Anzahl* & - & & Cursor aufwärts zum Zeilenanfang *Anzahl* Zeilen (BEW)

*Anzahl* & ) & & Cursor *Anzahl* Sätze vorwärts (BEW)

*Anzahl* & ( & & Cursor *Anzahl* Sätze zurück (BEW)

*Anzahl* & } & & Cursor *Anzahl* Absätze vorwärts (BEW)

*Anzahl* & { & & Cursor *Anzahl* Absätze zurück (BEW)

*Anzahl* & ^D & & bewegt den Cursor Richtung Dateiende um *Anzahl* Zeilen (Standard: 1/2 Bildschirmseite)

*Anzahl* & ^U & & bewegt den Cursor Richtung Dateianfang um *Anzahl* Zeilen (Standard: 1/2 Bildschirmseite)

& ^F & & bewegt den Cursor seitenweise zum Dateiende (BEW)

& ^B & & bewegt den Cursor seitenweise zum Dateianfang (BEW)

*Anzahl* & ^E & & scrollt in Richtung Dateiende *Anzahl* Zeilen

*Anzahl* & ^Y & & scrollt in Richtung Dateiende *Anzahl* Zeilen

& ` & *Taste* & Cursor zu dem mit *Taste* markierten Zeichen (BEW)

& ' & *Taste* & Cursor zu der mit *Taste* markierten Zeile (BEW)

*Anzahl* & G & & Cursor zur Zeile *Anzahl* (Standard: zur letzten Zeile) (BEW)

*Anzahl* & H & & Cursor zur *Anzahl*-ten Bildschirmzeile (Standard: oberste Zeile) (BEW)

& M & & positioniert den Cursor auf den Anfang der mittleren Bildschirmzeile (BEW)

*Anzahl* & L & & positioniert den Cursor auf den Anfang der letzten Bildschirmzeile (BEW)

& z & *Taste* & bewegt die aktuelle Zeile zum Anfang(+) Ende(-) oder zur Mitte(.) des Bildschirms (BEW)

## Kommandos zum Ändern von Text:

Befehl	Beschreibung
<i>d</i>	<i>Bereich</i> löscht den mit <i>Bereich</i> angegebenen Text (EDIT)
<i>D</i>	löscht den Text bis Zeilenende (EDIT)
<i>Anzahl</i> <i>x</i>	löscht das Zeichen, <b>auf</b> dem sich der Cursor befindet (EDIT)



<b>Anzahl</b> X		löscht <i>Anzahl</i> Zeichen links vom Cursor (EDIT)
>	<i>Bereich</i>	schiebt den Text nach rechts (EDIT)
<	<i>Bereich</i>	schiebt den Text nach links (EDIT)
<b>Anzahl</b> i	<i>eing</i>	fügt Text vor dem Cursor ein (EDIT)
<b>Anzahl</b> a	<i>eing</i>	fügt Text hinter dem Cursor ein (EDIT)
<b>Anzahl</b> I	<i>eing</i>	fügt Text am Zeilenanfang ein (EDIT)
<b>Anzahl</b> A	<i>eing</i>	hängt Text an das Zeilenende an (EDIT)
<b>Anzahl</b> o	<i>eing</i>	eröffnet unter der aktuellen Zeile eine neue und fügt Text ein (EDIT)
<b>Anzahl</b> O	<i>eing</i>	eröffnet über der aktuellen Zeile <i>Anzahl</i> neue und fügt Text ein (EDIT)
R	<i>eing</i>	überschreibt vorhandenen Text mit <i>eing</i> (EDIT)
<b>Anzahl</b> r	<i>Taste</i>	ersetzt <i>Anzahl</i> Zeichen durch <i>Taste</i> (EDIT)
<b>Anzahl</b> s	<i>eing</i>	ersetzt <i>Anzahl</i> Zeichen durch <i>eing</i> (EDIT)
c	<i>Bereich</i>	ändert den Text im Bereich <i>Bereich</i> (EDIT)
C	<i>eing</i>	ändert den Text bis Zeilenende in <i>eing</i> (EDIT)
<b>Anzahl</b> S	<i>eing</i>	ändert <i>Anzahl</i> Zeilen (wie <i>Anzahl</i> cc) (EDIT)
<b>Anzahl</b> i	<i>eing</i>	fügt Text vor dem Cursor ein (EDIT)
<b>Anzahl</b> a	<i>eing</i>	fügt Text hinter dem Cursor ein (EDIT)
<b>Anzahl</b> I	<i>eing</i>	fügt Text am Zeilenanfang ein (EDIT)
<b>Anzahl</b> A	<i>eing</i>	hängt Text an das Zeilenende an (EDIT)
\	<i>Bereich</i>	öffnet ein Pop-Up Menü zum Ändern von Text (ERW)
u		nimmt das letzte EDIT Kommando zurück
U		nimmt alle letzten Änderungen der momentanen Zeile zurück
<b>Anzahl</b> J		hängt die nachfolgenden <i>Anzahl</i> Zeilen an die momentane an (EDIT)

## Arbeiten mit Puffern und Marken:

### Befehl Beschreibung

& y & *Bereich* & kopiert den mit *Bereich* angegebenen Text in einen Puffer

*Anzahl* & Y & & kopiert die aktuelle und *Anzahl* weitere Zeilen in einen Puffer

& `` & *Taste* & legt fest, welcher Zwischenpuffer als nächstes verwendet wird

& p & & fügt ausgeschnittenen Text hinter dem Cursor ein (EDIT)

& P & & fügt ausgeschnittenen Text vor dem Cursor ein (EDIT)

& @ & *Taste* & interpretiert den Inhalt eines Zwischenspeichers als `vi' Befehle und führt sie aus

& *m* & *Taste* & markiert eine Zeile oder ein Zeichen

& *v* & & setzt den Anfang einer Markierung für *c*, *d*, *y*, *<*, *>*, *!*, *\* (ERW)

& *V* & & setzt den Anfang einer Zeilenmarkierung für *c*, *d*, *y*, *<*, *>*, *!*, *\* (ERW)

## Kommandos zum Suchen nach Ausdrücken und Wörtern:

Befehl	Beschreibung
--------	--------------

& ^A &	& sucht nach dem nächsten Vorkommen des Wortes, auf dem der Cursor steht (BEW) (ERW)
--------	--

*Anzahl* & % & & plaziert den Cursor auf die zugehörige ( ) { } [ ], oder plaziert den Cursor auf *Anzahl* Prozent der Datei (BEW) (ERW)

& / & *Text* & sucht vorwärts nach einem regulären Ausdruck *Text* (BEW)

& ? & *Text* & sucht rückwärts nach dem regulären Ausdruck *Text* (BEW)

## Wiederholungs Kommandos:

*Anzahl* & . & & wiederholt das vorhergehende EDIT Kommando

& *n* & & wiederholt das letzte Suchen (BEW)

& *N* & & wiederholt das letzte Suchen in umgekehrter Richtung (BEW)

*Anzahl* & ; & & wiederholt das vorhergehende *f*, *F*, *t*, *T* Kommando (BEW)

*Anzahl* & , & & wiederholt das vorhergehende *f*, *F*, *t*, *T* Kommando in umgekehrter Richtung (BEW)

*Anzahl* & & & & wiederholt den letzten *:s//* Befehl an der momentanen Position (EDIT)

## Sonstige Kommandos:

Befehl	Beschreibung
--------	--------------

& ^G &	& zeigt den Dateistatus und die momentane Zeilennummer
--------	--

& Z & Z & speichert die Datei und beendet *elvis*

& ^L & & erneuert den Bildschirm (wie ^R)

& ^R & & erneuert den Bildschirm (wie ^L)

& ! & *Bereich* & führt die selektierten Zeilen einem externen Filter zu, der nach dem Kommando angegeben werden muß

& = & *Bereich* & formatiert *Bereich* neu

& ^^ & & zur vorherigen Datei

*Anzahl* & # & + & erhöht eine Zahl (EDIT) (ERW)

& : & *Text* & führt ein einzelnes `ex' Kommando aus

& *Q* & & schaltet in den `ex' Modus

& \* & & bewegt den Cursor zum nächsten Fehler in der Fehlerliste (ERW)

& *K* & & dient in Verbindung mit dem Programm *ctags* dazu, Schlüsselworte nachzuschlagen

& ^] & & wenn sich der Cursor auf einem tag-Namen befindet, gehe dorthin

## Eingabemodi

Im `visual mode' läßt sich Text **nicht** eingeben, sondern es muß erst in einen sog. `input mode' geschaltet werden. Dies geschieht mit den Befehlen *A*, *C*, *I*, *O*, *R*, *S*, *a*, *i*, *o*, *s*. Die *S*, *s*, *C*, *c* Kommandos setzen zeitweilig ein `\$' an das Ende des Textbereiches, auf den sie wirken. Im `input mode' werden alle Zeichen mit Ausnahme der folgenden in den Text eingefügt.

Befehl & Funktion

^*A* & fügt eine Kopie der letzten Texteingabe ein

^*D* & löscht ein Einrückungszeichen

^*H* & löscht das Zeichen vor dem Cursor

^*L* & baut den Bildschirm neu auf

^*M* & (Wagenrücklauf) fügt eine neue Zeile ein (^*J*, Zeilenvorschub)

^*O* & führt den nächsten Tastendruck als `visual mode' Befehl aus (eingeschränkt)

^*P* & fügt den Inhalt des Zwischenpuffers ein

^*R* & wie ^*L*

^*T* & fügt ein Einrückungszeichen ein

^*U* & löscht alle eingegebenen Zeichen bis zum Zeilenanfang

^*V* & fügt den folgenden Tastendruck ein, auch wenn er eine Sonderbedeutung hat

^*W* & löscht alle eingegebenen Zeichen bis zum Wortanfang

^*Z*^*Z* & speichert die Datei und beendet *elvis*

^[ & (ESCAPE) schaltet in den `visual command mode' zurück

Auf einigen Systemen kann mit ^*S* und ^*Q* die Bildschirmausgabe angehalten und wieder gestartet, oder mit ^*C* *elvis* abgebrochen werden. ^@ (das NULL Zeichen) kann nicht in den Text eingefügt werden.

Der `visual mode' Befehl *R* schalten in den sog. `replace mode'. In diesem Modus werden vorhandene Zeichen durch die eingegebenen ersetzt. Ist das Zeilenende erreicht, werden die folgenden Eingaben an die Zeile angefügt.

## Die Cursortasten in den Eingabemodi

Im Gegensatz zu *ex/vi* ist es bei *elvis* möglich, in den Eingabemodi die Cursortasten zu verwenden. Desweiteren funktionieren auch die BILDAUF, BILDAB, POS1 und ENDE Tasten. Die ENTf Taste löscht ein einzelnes Zeichen im `input mode' und die EINFg Taste schaltet zwischen `input mode' und `replace mode' um. Der Hauptvorteil dabei ist, daß sich *elvis*, solange man sich im `input mode' befindet, fast wie jeder normale Editor verhält. Bei fast allen anderen Editoren befindet sich der Benutzer immer im `input' oder `replace mode' und kann die Cursortasten jederzeit verwenden. Zusammen mit der ^*Z* ^*Z* Befehlsfolge braucht man für einfache Änderungen nie in den `visual command mode' zu schalten. Leider scheint *elvis* bezüglich der Funktion dieser Sondertasten, hohe Ansprüche an die Eintragungen

in der Datei */etc/termcap* zu stellen. Deshalb bedarf es auf vielen Systemen einer Anpassung dieser Datei.

## Digraphs

Ein Digraph ist ein Zeichen, das aus zwei anderen zusammengesetzt ist. *elvis* unterstützt Digraphs als Möglichkeit nicht-ASCII Zeichen einzugeben. So sollte z. B. ein Apostroph und ein `e' als ein *é* angezeigt und gespeichert werden.

Bedauerlicherweise existiert kein Standard für erweiterte ASCII-Zeichen. *elvis* kann so übersetzt werden, daß er entweder den PC-, den ISO-LATIN1-Zeichensatz, der vom X11-Window System verwendet wird, oder keinen von beiden unterstützt. Die Digraph-Tabelle kann mit dem `colon mode' Befehl *:digraph* angezeigt und verändert werden. Bevor nicht das Kommando *:set digraph* eingegeben wurde, erkennt *elvis* Digraphs nicht. Um ein Digraph einzugeben, muß zuerst das eine Zeichen, dann *^H* (BACKSPACE), dann das andere Zeichen eingegeben werden. *elvis* ersetzt dann das letzte Zeichen durch das zusammengesetzte.

## Abkürzungen

*elvis* kann Abkürzungen erweitern. Mit dem `colon mode' Kommando *:abbr* kann eine Abkürzung definiert werden. Wird dann im `input mode' die Abkürzung eingegeben, erweitert sie *elvis* sofort auf den vollen Ausdruck. *elvis* führt die Erweiterung durch, sobald das erste nicht alphanumerische Zeichen eingegeben wird. Um die Ersetzung zu verhindern, kann vor dem ersten nicht alphanumerischen Zeichen *^V* eingegeben werden.

## Automatisches Einrücken

Wird mit der Option *:set autoindent* das automatische Einrücken eingeschaltet, fügt *elvis* vor jedem Zeilenanfang automatisch soviel Leerraum ein, wie in der vorangegangenen Zeile verwendet wurde. Der Leerraum am Zeilenanfang läßt sich mit *^T* vergrößern und mit *^D* verkleinern. Wird die Option *:set noautotab* verwendet, fügt *elvis* statt Tabulatorzeichen Leerzeichen ein. Der `auto indent' Modus von *elvis* ist mit dem von *ex/vi* nicht 100%ig identisch. *0^D* und *^^D* funktionieren nicht, *^U* löscht den gesamten Leerraum, und in einigen Situationen fügt *elvis* weniger oder mehr Leerraum ein, als *ex/vi*.

## Der `colon mode'

### `Colon mode' Befehle

Zeilen    Befehl    Argumente

& *ab[br]* & [*Kurzform*] [*erweiterteForm*]

[*Zeile*] & *a[ppend][!]* &

& *ar[gs]* & [*Dateien*]

& *cc* & [*Dateien*]

*s[ubstitute] / regAusdruck/Ersatz/[p][g][c]*

	<i>ta[g][!]</i>	<i>Tagname</i>
	<i>una[bbr]</i>	<i>[Abkürzung]</i>
	<i>u[ndo]</i>	
	<i>unm[ap][!]</i>	<i>Taste</i>
	<i>ve[rsion]</i>	
<i>[Zeile][,Zeile]</i>	<i>v[global]</i>	<i>/ regAusdruck/Befehl</i>
	<i>vi[sual]</i>	<i>Dateiname</i>
	<i>wq</i>	
<i>[Zeile][,Zeile]</i>	<i>w[rite][!]</i>	<i>[[&gt;&gt;]Datei]</i>
	<i>x[it][!]</i>	
<i>[Zeile][,Zeile]</i>	<i>y[ank]</i>	<i>x</i>
<i>[Zeile][,Zeile]</i>	<i>!</i>	<i>externerBefehl</i>
<i>[Zeile][,Zeile]</i>	<i>&lt;</i>	
<i>[Zeile][,Zeile]</i>	<i>=</i>	
<i>[Zeile][,Zeile]</i>	<i>&gt;</i>	
<i>[Zeile][,Zeile]</i>	<i>&amp;</i>	
	<i>@</i>	<i>x</i>

Um **`colon mode'** Kommandos zu verwenden, muß vom **`visual mode'** in den **`colon mode'** umgeschaltet werden. Dies geschieht mit **`:'** für ein Kommando und mit **Q** bis zum nächsten **:vi** Kommando.

## Zeilenangaben

Zeilenangaben sind immer optional. Die erste Zeilenangabe wird normalerweise als die momentane Zeile angenommen, die zweite wird gleich der ersten gesetzt. Ausnahmen hiervon sind die Befehle **:write**; **:global** und **:vglobal**, die sich, wenn nicht anders angegeben, auf den gesamten Text beziehen und **:/**, der standardmäßig auf keine Zeile wirkt. Zeilenangaben bestehen aus einem absoluten und/oder einem relativen Teil.

Der absolute Teil einer Zeilenangabe kann eine Zeilennummer, eine Marke, ein **`.'**, um die aktuelle Zeile zu bezeichnen, ein **\$** um die letzte Zeile des Textes zu bezeichnen oder ein vorwärts oder rückwärts Suchen sein. Eine Zeilennummer ist eine Ziffernfolge, die als Dezimalzahl interpretiert wird. Eine Marke wird als ein **'** gefolgt von einem Buchstaben angegeben. Marken müssen gesetzt werden, bevor sie verwendet werden können. Im **`visual mode'** kann eine Marke mit dem Befehl **m** gefolgt von einem Buchstaben gesetzt werden. Im **`colon mode'** wird eine Marke mit dem **:mark** Befehl gesetzt. Ein vorwärts Suchen wird als ein von **/** eingeschlossener regulärer Ausdruck angegeben. Das Suchen beginnt in der aktuellen Zeile. Ein rückwärts Suchen wird als ein von **?** eingeschlossener regulärer Ausdruck angegeben. Das Suchen beginnt in der vorhergehenden Zeile.

Der relative Teil einer Zeilenangabe wird als **+** oder **-** gefolgt von einer Dezimalzahl angegeben. Die

Zahl wird von dem absoluten Teil abgezogen oder hinzuaddiert.

Ein Sonderfall ist das % Zeichen. Es wird dazu verwendet, alle Zeilen des Textes zu bezeichnen (es ist mit 1,\$ identisch).

Beispiele:

<b>:p</b>	gibt die aktuelle Zeile aus
<b>:37p</b>	gibt Zeile 37 aus
<b>:'gp</b>	gibt die Zeile, in der die Marke <i>g</i> gesetzt ist aus
<b>:/foo/p</b>	gibt die nächste Zeile, die <i>foo</i> enthält aus
<b>:\$p</b>	gibt die letzte Zeile des Textes aus
<b>:20,30 p</b>	gibt die Zeilen 20-30 aus
<b>:1,\$p</b>	gibt den gesamten Text aus
<b>:%p</b>	gibt ebenfalls den gesamten Text aus
<b>:/foo/-2,+4p</b>	gibt 5 Zeilen um das nächste Vorkommen von <i>foo</i> aus

## Texteingabe Kommandos

[Zeile] **append**

[Zeile][,Zeile] **change**

[Zeile] **insert**

Das Kommando *append* fügt Text hinter der angegebenen Zeile ein.

Das Kommando *insert* fügt Text vor der angegebenen Zeile ein.

Das Kommando *change* kopiert die angegebenen Zeilen in einen Zwischenpuffer, löscht sie und fügt an ihrer Stelle neuen Text ein.

Bei diesen Kommandos wird das Eingeben durch ^D oder durch eine Zeile, die nur einen Punkt enthält, beendet.

## Ausschneiden und Einfügen

[Zeile][,Zeile] **delete** [x]

[Zeile][,Zeile] **yank** [x]

[Zeile] **put** [x]

[Zeile][,Zeile] **copy** Zeile

[Zeile][,Zeile] **to** Zeile

[Zeile][,Zeile] **move** Zeile

Das *delete* Kommando kopiert die angegebenen Zeilen in einen Zwischenpuffer, und löscht sie dann.

Das *yank* Kommando kopiert die angegebenen Zeilen in einen Zwischenpuffer, löscht sie aber nicht.

Das *put* Kommando fügt den Text eines Zwischenpuffers hinter der angegebenen Zeile ein. Bei diesen Kommandos ist *x* die optionale Angabe eines Zwischenpuffers, mit dem das Kommando arbeiten soll.

Die *copy* und *to* Kommandos kopieren den Text direkt hinter eine andere Zeile.

Das *move* Kommando löscht die angegebenen Zeilen und fügt sie sofort hinter einer anderen ein. Liegt die Zielzeile hinter den gelöschten Zeilen, so werden die Zeilennummern automatisch korrigiert.

## Kommandos zum Anzeigen von Text

[Zeile][,Zeile] **print**

[Zeile][,Zeile] **list**

[Zeile][,Zeile] **number**

Das *print* Kommando stellt die angegebenen Zeilen auf dem Bildschirm dar.

Das *list* stellt die Zeilen ebenfalls dar, zeigt jedoch auch Control-Zeichen an.

Das *number* Kommando zeigt die Zeilen mit Zeilennummern an.

## Kommandos, die auf den gesamten Text wirken

[Zeile][,Zeile] **global** / *regAusdruck/Befehl*

[Zeile][,Zeile] **vglobal** / *regAusdruck/Befehl*

Das *global* Kommando durchsucht die angegebenen Zeilen (oder die ganze Datei, wenn keine Zeilennummern angegeben wurden) nach dem regulären Ausdruck, positioniert den Cursor auf den entsprechenden Zeilen und führt *Befehl* darauf aus.

Das *vglobal* Kommando arbeitet im Prinzip genauso, nur führt es *Befehl* in allen Zeilen aus, die den regulären Ausdruck **nicht** enthalten.

## Kommandos zum Editieren einzelner Zeilen

[Zeile][,Zeile] **join**[!]

[Zeile][,Zeile] **!** *programm*

[Zeile][,Zeile] **<**

[Zeile][,Zeile] **>**

[Zeile][,Zeile] **substitute**/*regAusdruck/Ersatz*/[p][g][c]

[Zeile][,Zeile] **&**

Das *join* Kommando hängt alle angegebenen Zeilen aneinander. Wird nur eine Zeile angegeben, wird die darauffolgende Zeile angehängt. Der *join* Befehl fügt ein oder zwei Leerzeichen zwischen die Zeilen ein. Wird die Variante *:join!* verwendet, unterbleibt dies.

Das **!** Kommando führt die entsprechenden Zeilen einem externen Filterkommando zu und ersetzt sie durch die Ausgabe des Filters. So würde beispielsweise das Kommando *:a,z! sort* die Zeilen zwischen den Marken *a* und *z* alphabetisch sortieren.

Die **<** und **>** Kommandos rücken die Zeilen um die Größe eines Tabulatorzeichens ein oder aus. Die Größe des Tabulators wird durch die Option *shiftwidth* festgelegt.

Der Befehl *substitute* sucht nach dem regulären Ausdruck und ersetzt ihn durch *Ersatz*. Die *p* Option gibt die geänderten Zeilen aus, die *c* Option fragt vor jedem Ersetzen, ob das Ersetzen durchgeführt werden soll. Wenn die *g* Option nicht angegeben wird, bearbeitet *elvis* nur das erste Auftreten des



Ausdrucks in jeder Zeile.

Das `&` Kommando wiederholt das letzte Suchen und Ersetzen. Es sucht nach dem zuletzt eingegebenen Suchmuster (auch wenn es bei einem anderen Kommando angegeben wurde) und ersetzt es durch *Ersatz* des letzten *substitute* Befehls.

## Der undo Befehl

Das *undo* Kommando nimmt die Änderungen des letzten Editierkommandos zurück.

## Konfiguration und Status

```
map[!] [Taste soll_übersetzt_werden_zu]  
unmap[!] Taste  
abbr [Kurzform] [erweiterteForm]  
unabbr [Kurzform]  
digraph[!] [XX [Y]]  
set [Optionen]  
mkexrc  
[Zeile] mark x  
visual  
version  
[Zeile][Zeile] =  
file [Datei]  
source Datei  
@ x  
color [textart] [[bright] Farbe] [on Farbe]
```

Mit dem *map* Kommando kann *elvis* so konfiguriert werden, daß er Funktions- und Sondertasten erkennt und ihnen eine benutzerdefinierte Funktion zuweist. Normalerweise wird die Übersetzung nur im 'visual mode' durchgeführt. Wird statt `:map` aber `:map!` angegeben, so führt *elvis* diese Funktion auch im 'input' oder 'replace mode' aus. Wird `:map` kein Argument übergeben, gibt es eine Liste der momentan aktiven Übersetzungstabelle aus. Wird der Befehl mit zwei Parametern aufgerufen, so ist der erste die Zeichenfolge, die die Taste tatsächlich sendet, und der zweite die Tastenfolge, die von *elvis* ausgeführt werden soll. Ist das erste Argument eine Zahl, so übersetzt *elvis* sie zu der entsprechenden Funktionstaste. So würde `map 5 dd` der Taste F5 das Kommando `dd` zuweisen, d. h. eine Zeile löschen.

Das *unmap* Kommando nimmt die mit *map* festgelegten Tastaturdefinitionen für die entsprechende Taste wieder zurück.

Das *abbr* Kommando dient dazu, eine Abkürzungstabelle anzuzeigen bzw. zu erstellen. Die Tabelle besteht aus den gewünschten Abkürzungen und den dazugehörigen ausgeschriebenen Wörtern. Im 'input mode' ersetzt *elvis* *Kurzform* durch *erweiterteForm*, sobald nach *Kurzform* ein nicht alphanumerisches Zeichen eingegeben wird. Soll die Ersetzung verhindert werden, so muß als erstes Zeichen nach *Kurzform* ein `^V` eingegeben werden. Ohne Parameter gibt das Kommando die Tabelle aus. Mit zwei Parametern wird der erste als Abkürzung und der Rest der Zeile als ausgeschriebene Form betrachtet. Das *unabbr* Kommando löscht Einträge aus der Abkürzungstabelle.

Das *digraph* Kommando erlaubt es dem Benutzer, die von *elvis* verstandenen Digraphs anzuzeigen,

zu erweitern oder einzelne Digraphs aus der Tabelle zu entfernen. Ohne Parameter wird die Tabelle angezeigt. Um ein Digraph zu setzen, müssen dem Befehl zwei Parameter übergeben werden. Der erste sind die beiden Zeichen, aus denen das Digraph zusammengesetzt ist, der zweite ist das nicht-ASCII Zeichen, das durch die beiden ersten dargestellt werden soll. Das höchstwertigste Bit des nicht-ASCII Zeichens wird von dem *digraph* Kommando automatisch gesetzt, wenn kein *!* an den Kommandonamen angehängt wird. Wird dem Kommando nur der erste Parameter übergeben, so wird das entsprechende Zeichen aus der Tabelle gelöscht.

Das *set* Kommando dient zum Setzen und Anzeigen der *elvis*-Optionen. Ohne Argumente zeigt es die geänderten Optionen an. Mit dem Argument *all* zeigt es den Zustand aller Optionen an. Ansonsten wird das Argument als eine Option, die gesetzt werden soll, behandelt.

Der Befehl *mkexrc* speichert die momentane Konfiguration in einer Datei names *`.exrc'* im aktuellen Verzeichnis ab.

Das *mark* Kommando setzt an die angegebene Stelle eine Marke. Sie kann später dazu verwendet werden, um in einem Kommando eine Zeile zu referenzieren.

Das *visual* Kommando schaltet aus dem *`.colon mode'* in den *`.visual mode'* zurück.

Der *version* Befehl gibt die Versionsnummer von *elvis* aus.

Das *=* Kommando gibt aus, welche Zeile angegeben wurde oder, wenn ein Bereich angegeben wurde, den Anfangs- sowie den Endpunkt und die Anzahl der Zeilen, die dazwischen liegen.

Das Kommando *file* gibt den Dateinamen, die Anzahl der Zeilen und den Veränderungsstatus aus. Es kann auch dazu verwendet werden, den Dateinamen zu ändern.

Das *source* Kommando liest eine Folge *`.colon mode'* Befehle aus einer Datei ein und führt die Befehle aus.

Das *@* Kommando führt den Inhalt eines Zwischenspeichers als Befehle aus.

Der *color* Befehl funktioniert nur unter MS-DOS oder auf einem ANSI-Farbterminal. Er ermöglicht verschiedene Vorder- und Hintergrundfarben für verschiedene Texttypen (normal, fett, kursiv, das PopUp-Menü und die sichtbaren Markierungen) festzulegen. Standardmäßig ändert es die *`.normal'* Farben. Um die Farben für andere Textdarstellungen zu ändern, muß als erstes Argument der Anfangsbuchstabe des Texttyps angegeben werden (z. B. *`.color bright yellow on blue'* ändert die Standardtextdarstellung in hellgelben Text auf blauem Hintergrund; *`.color b bright white'* ändert die Textdarstellung von Fettdruck in hellweiße Schrift auf blauem Hintergrund). Die Hintergrundfarbe entspricht, wenn nicht angegeben, immer der momentanen Hintergrundfarbe. Nur in dem ersten *:color* Befehl muß sowohl die Vorder- als auch die Hintergrundfarbe für die normale Textdarstellung angegeben werden.

## Kommandos zum Arbeiten mit mehreren Dateien

**args** [*Dateien*]  
**next** [*!]* [*Dateien*]  
**Next** [*!]*  
**previous** [*!]*  
**rewind** [*!]*

Wenn *elvis* von der Kommandozeile der Shell gestartet wird, werden alle Dateinamen, die übergeben

werden, in der Kommandozeilenliste gespeichert. Das `:args` Kommando zeigt die Kommandozeilenliste an oder definiert eine neue.

Das `:next` Kommando wechselt von einer Datei in der Kommandozeilenliste zur nächsten. Auch hier kann eine neue Kommandozeilenliste angegeben werden.

Das `:Next` und das `:previous` Kommando (sie sind völlig gleichbedeutend) schalten zur vorhergehenden Datei.

Das `:rewind` Kommando schaltet zur ersten Datei in der Kommandozeilenliste.

## Zwischen Dateien wechseln

**edit**[!] *Datei*  
**tag**[!] *Tagname*

Das `:edit` Kommando dient zum Wechseln der Datei. Es hat nichts mit der Kommandozeilenliste zu tun.

Der `:tag` Befehl sucht den angegebenen Referenzpunkt *Tagname* in einer Datei namens ``tags'`. Diese Datei enthält eine Liste der verfügbaren Referenzpunkte und der Dateien, in denen sie sich befinden. *elvis* wechselt dann zu der Datei und plazierte den Cursor auf dem Referenzpunkt. Eine solche ``tags'`-Datei wird beispielsweise von dem Programm ``ctags'` erstellt.

## Arbeiten mit einem Compiler

**cc** [*Dateien*]  
**make** [*Ziel*]  
**errlist**[!] [*Fehlerliste*]

Das `:cc` und das `:make` Kommando starten den Compiler oder das *make*-Dienstprogramm und leiten deren Ausgabe in eine Fehlerdatei namens *Fehlerliste* um. Werden keine Dateien angegeben, wird dem Compiler die aktuelle Datei übergeben (sie muß vorher gespeichert werden). Der Inhalt der Fehlerdatei wird dann nach Fehlermeldungen durchsucht. Wird eine Fehlermeldung gefunden, wechselt *elvis* zu der Datei, in der der Fehler aufgetreten ist und plazierte den Cursor in der entsprechenden Zeile. In der Statuszeile von *elvis* wird die Fehlerbeschreibung angezeigt. Wurde ein Fehler behoben, wird der Cursor auf die nächste Zeile, in der ein Fehler aufgetreten ist, gesetzt. Im ``visual mode'` geschieht das auch durch das ``*' Kommando.`

Es ist auch möglich außerhalb von *elvis* eine Fehlerdatei zu erstellen und *elvis* mit der `-m` Option zu starten. Der Cursor wird dann ebenfalls auf die Zeile gesetzt, in der der erste Fehler aufgetreten ist. Da in der Fehlerdatei gespeichert ist, in welcher Datei der Fehler aufgetreten ist, muß kein Dateinamen angegeben werden.

Wird das `:errlist` Kommando wiederholt ausgeführt, so versucht *elvis* die Änderungen der Zeilennummern durch das Einfügen oder Löschen von Zeilen zu berücksichtigen. Diese Korrekturen werden in der Annahme durchgeführt, daß die Datei vom Anfang zum Ende durchgearbeitet wird.

## elvis beenden

```
quit[!]  
wq  
xit
```

Das `:quit` Kommando beendet *elvis* ohne die Datei zu speichern.

Das `:wq` Kommando speichert die Datei und beendet danach *elvis*.

Das `:xit` Kommando arbeitet wie das `wq` Kommando, außer daß es die Datei nur speichert, wenn sie geändert wurde.

## Datei Ein- und Ausgabekommandos

```
[Zeile] read Datei  
[Zeile][,Zeile] write[!] [[>>]Datei]
```

Das `:read` Kommando liest Text aus einer anderen Datei ein und fügt ihn hinter der angegebenen Zeile an. Es kann ebenfalls die Ausgabe eines anderen Kommandos einlesen, indem dem Programmnamen ein `!` vorangestellt und anstelle von *Datei* verwendet wird.

Das `write` Kommando schreibt die gesamte Datei oder einen Teil in eine andere. Wird `!` angegeben, so wird die Datei geschrieben, selbst wenn das *readonly*-Flag gesetzt ist. Wenn dem Dateinamen ``>>'` vorangestellt wird, werden die Zeilen an die Datei angehängt. Die Ausgabe des `write` Befehls kann der Standardeingabe eines Programms zugeführt werden, wenn statt des Dateinamens der Programmname mit vorangestelltem `!` angegeben wird. Vorsicht: zwischen dem Ausrufezeichen und dem Programmnamen muß mindestens ein Leerzeichen stehen (*w!Dateiname*, aber *w! Programmname*).

## Verzeichnis Kommandos

```
cd [Verzeichnis]  
chdir [Verzeichnis]  
shell
```

Die Kommandos `:cd` und `:chdir` wechseln das aktuelle Arbeitsverzeichnis (synonym).

Das Kommando `:shell` startet eine interaktive Shell.

## Debugging Kommandos

```
[Zeile][,Zeile] debug[!]  
validate[!]
```

Diese Kommandos stehen nur zur Verfügung, wenn *elvis* mit der *-DDEBUG* Option übersetzt wurde.

Das `:debug` Kommando gibt die Daten des Blockes aus, der die angegebenen Zeilen enthält. Wird `!` mit angegeben, so wird zusätzlich noch der Inhalt des Blockes angezeigt.

Das `:validate` Kommando überprüft bestimmte interne Variablen auf ihre Konsistenz. Normalerweise produziert das Kommando keine Ausgabe, wenn es keine Fehler entdeckt. Wird `!` angegeben, gibt es immer ``etwas'` aus.

# Reguläre Ausdrücke

*elvis* kann zum Suchen und Ersetzen reguläre Ausdrücke verwenden. Ein regulärer Ausdruck ist eine Zeichenfolge, in der einige Zeichen eine Sonderbedeutung haben. Durch die Verwendung regulärer Ausdrücke bietet sich die Möglichkeit, sehr komplizierten Suchaufgaben gerecht zu werden.

## Funktion

*elvis'* Funktion zum Auswerten regulärer Ausdrücke belegt die folgenden Zeichen (Metazeichen genannt) mit einer Sonderbedeutung.

**\(unterAusdruck)\**

Die Metazeichen `\(` und `\)` werden dazu verwendet, Unterausdrücke in regulären Ausdrücken zu begrenzen. Trifft der reguläre Ausdruck auf einen Textbereich zu, so behält *elvis* auf welchen Teilbereich *unterAusdruck* zutrifft. Das Kommando `s/regAusdruck/neuerText/` benutzt diese Funktion.

**^**

Das `^` Metazeichen bezeichnet einen Zeilenanfang. Soll z. B. *foo* am Zeilenanfang gefunden werden, so ist der nötige reguläre Ausdruck `/^foo/`. `^` ist nur ein Metazeichen, wenn es am Anfang eines regulären Ausdrucks steht.

**\$**

Durch `$` wird in einem regulären Ausdruck ein Zeilenende bezeichnet. `$` ist nur ein Metazeichen, wenn es am Ende eines regulären Ausdrucks steht. `/$$/` würde demnach auf ein `$` Zeichen am Zeilenende zutreffen.

**\<**

Das `\<` Metazeichen findet eine Zeichenkette der Länge Null am Anfang eines Wortes. Eine Zeichenkette, die aus mindestens einem Buchstaben oder einer Zahl besteht, wird als Wort betrachtet. Ein Wort kann nach jedem nicht alphanumerischen Zeichen beginnen.

**\>**

Das `\>` Metazeichen findet eine Zeichenkette der Länge Null am Ende eines Wortes. Ein Wort endet am Zeilenende oder vor einem nicht alphanumerischen Zeichen. Der reguläre Ausdruck `\<ende\>/` würde jedes Vorkommen des Wortes *ende* im Text finden, ohne jedoch auch das Vorkommen von *ende* innerhalb eines anderen Wortes (z. B. *Kal ender*) anzuzeigen.

**.**

Das Metazeichen ``.'` steht für jedes beliebige Zeichen.

**[zeichenliste]**

Dieser Ausdruck trifft auf jedes Zeichen zu, das in *zeichenliste* enthalten ist. In *zeichenliste* kann ein Zeichenbereich angegeben werden, indem zwischen zwei Zeichen ein `-` gesetzt wird. `[a-zA-Z]` würde auf jeden Buchstaben zutreffen. Wird der *zeichenliste* ein `^` vorangestellt, so trifft sie jedes Zeichen, das nicht in ihr enthalten ist. `[^ ]` würde alle Zeichen außer dem Leerzeichen bezeichnen.

**\{n\}**

Dies ist ein Näherungsoperator, d. h. er kann nur nach einem Ausdruck angegeben werden, der ein einzelnes Zeichen liefert. Er gibt an, wie oft der Ausdruck, der das Zeichen zurückliefert,

wiederholt werden soll.  $n$  ist hierbei die Anzahl der Wiederholungen.  $/^-\{80\}/$  bezeichnet eine Zeile aus 80 Bindestrichen.  $/\<[a-zA-Z]\{4\}/$  bezeichnet jedes aus vier Buchstaben bestehende Wort.

$\{n,m\}$

Dies ist ein Näherungsoperator, d. h. er kann nur nach einem Ausdruck angegeben werden, der ein einzelnes Zeichen liefert. Er gibt an, wie oft der Ausdruck, der das Zeichen zurückliefert, wiederholt werden soll.  $n,m$  ist hierbei der Bereich, indem die Anzahl der Wiederholungen liegen muß. Wird  $m$  weggelassen (das Komma muß bleiben), so wird für  $m$  'unendlich' eingesetzt.  $/\<[a-zA-Z]\{4,6\}/$  bezeichnet jedes aus vier, fünf oder sechs Buchstaben bestehende Wort.

\*

Dies ist ein Näherungsoperator, d. h. er kann nur nach einem Ausdruck angegeben werden, der ein einzelnes Zeichen liefert und bezeichnet eine beliebige Anzahl Wiederholungen. Er ist gleichbedeutend mit  $\{0,\}$ .

$\{+\}$

Dies ist ein Näherungsoperator, d. h. er kann nur nach einem Ausdruck angegeben werden, der ein einzelnes Zeichen liefert und bezeichnet eine beliebige Anzahl Wiederholungen, mindestens jedoch eine. Er ist gleichbedeutend mit  $\{1,\}$ .

$\{?\}$

Dies ist ein Näherungsoperator, d. h. er kann nur nach einem Ausdruck angegeben werden, der ein einzelnes Zeichen liefert und drückt aus, daß der vorhergehende Ausdruck optional ist. Er ist gleichbedeutend mit  $\{0,1\}$ .

Alle anderen Zeichen werden nicht interpretiert und müssen dem Text exakt entsprechen. Um die Sonderbedeutung der Metazeichen aufzuheben, muß ihnen ein  $\backslash$  vorangestellt werden.

## Ersetzungen

Das `:s` Kommando benötigt mindestens zwei Argumente: einen regulären Ausdruck, und eine Zeichenkette, durch die der Text, auf den der reguläre Ausdruck zutrifft, ersetzt werden soll. Auch in der Ersetzungszeichenkette haben einige Zeichen eine Sonderbedeutung.

$\&$

fügt eine Kopie des Originaltextes ein

$\sim$

(Tilde) fügt eine Kopie des vorhergehenden Ersetzungstextes ein

$\backslash x$

fügt eine Kopie des Originaltextes ein, auf den der  $x$ -te Unterausdruck  $\backslash(\backslash)$  zutrifft

$\backslash U$

konvertiert den Text der nächsten & und \x Kommandos in Großbuchstaben

**\L**

konvertiert den Text der nächsten & und \x Kommandos in Kleinbuchstaben

**\E**

hebt die Wirkung des letzten \U oder \L Kommandos wieder auf

**\u**

konvertiert das erste Zeichen des nächsten & oder \x in einen Großbuchstaben

**\l**

konvertiert das erste Zeichen des nächsten & oder \x in einen Kleinbuchstaben

Um die Sonderbedeutung der Metazeichen aufzuheben, muß ihnen ein \ vorangestellt werden. Ist die Option `nomagic' gesetzt, so haben & und ~ keine Sonderbedeutung, außer ihnen wird ein \ vorangestellt.

## Optionen

*elvis* besitzt zwei Optionen, mit denen die Art, in der reguläre Ausdrücke verwendet werden, gesteuert werden kann. Die erste *[no]magic* ist eine boolean Option, die standardmäßig wahr ist. Solange diese Option eingeschaltet ist, haben alle Metazeichen die oben beschriebenen Bedeutungen. Wird die Option mit *:set nomagic* auf falsch gesetzt, so behalten nur ^ und \$ ihre Sonderbedeutung. Die andere Option ist ebenfalls eine boolean Variable, und heißt *[no]ignorecase*. Sie ist standardmäßig auf falsch gesetzt. Wird diese Option gesetzt, so unterscheidet *elvis* bei Suchkommandos nicht zwischen Klein- und Großbuchstaben.

## Beispiele

Dieses Beispiel ändert jedes Vorkommen von `Egon' in `Fritz':

```
:%s/Egon/Fritz/g
```

Dieses Beispiel löscht den Leerraum am Zeilenende jeder Zeile (die eckigen Klammern enthalten ein Leerzeichen und ein Tabulatorzeichen):

```
:%s/[ ]\+$//
```

Dieses Beispiel ändert alle Buchstaben in der aktuellen Zeile in Großbuchstaben:

```
:s/.*/\U&/
```

Dieses Beispiel unterstreicht alle Buchstaben der aktuellen Zeile, indem es sie in mit Hilfe eines Rückschritts unterstrichene Buchstaben umwandelt:

```
:s/[A-Za-z]/_ ^H&/g
```

Dieses Beispiel sucht den letzten Doppelpunkt einer Zeile und vertauscht den Text vor dem Doppelpunkt mit dem hinter dem Doppelpunkt. Das erste `\( \)` Paar dient dazu, den Text vor dem Doppelpunkt einzulesen, das zweite liest den Text hinter dem Doppelpunkt. Die Metazeichen `\ 1` und `\ 2` werden in umgekehrter Reihenfolge eingegeben, um die Zeile am Doppelpunkt gespiegelt wieder auszugeben:

```
:s/\(.*\):\(.*\)/\ 2:\ 1/
```

## Die Optionen von elvis

Die Optionen werden mit dem 'colon mode' Kommando `:set` gesetzt. Der Wert der Optionen beeinflusst das Verhalten der nachfolgenden Befehle. Der Bequemlichkeit zuliebe haben die Optionen einen langen Name, der ihre Bedeutung beschreibt, und einen kurzen Namen, der schnell einzugeben ist. *elvis* versteht beide Namen. Es gibt drei verschiedene Arten Optionen: boolean, Zeichenketten und numerische. Boolean Optionen werden auf wahr gesetzt, indem ihr Name dem `:set` Kommando übergeben wird. Wird ihrem Namen ein *no* vorangestellt (z. B. *nomagic*) so werden sie auf falsch gesetzt. Zusätzlich gibt es die Möglichkeit, boolean Optionen ein *neg* voranzustellen, wodurch ihr aktueller Status umgekehrt wird. Dies ist eine Erweiterung von *elvis* gegenüber *ex/vi*. Um den Inhalt einer numerischen oder einer Zeichenkettenoption zu ändern, gibt man im `:set` Kommando den Namen gefolgt von einem '=' und dem neuen Wert an (z. B. `:set tabstop=8`). Bei Zeichenkettenoptionen kann der Wert in Anführungszeichen eingeschlossen werden.

### **autoindent (ai)**

Wird die *autoindent* Option gesetzt, rückt *elvis* jede neue Zeile soweit wie die vorherige ein. Ohne diese Option beginnt jede Zeile in der ersten Spalte.

### **autoprint (ap)**

Diese Option betrifft nur den 'ex mode'. Ist diese Option gesetzt, gibt *elvis*, wenn der Cursor in eine neue Zeile bewegt wird oder das letzte Kommando die Datei verändert hat, die aktuelle Zeile aus.

### **autotab (at)**

Die Option bestimmt das Verhalten von *elvis* beim Einfügen von Leerraum an Zeilenanfang (*autoindent*). Ist die *autotab* Option gesetzt, so verwendet *elvis* eine Mischung aus Tabulatorzeichen und Leerzeichen um die richtige Menge Leerraum einzufügen. Ist die Option ausgeschaltet, so verwendet *elvis* nur Leerzeichen. Die *autotab* Option betrifft nur den automatisch eingefügten Leerraum.

### **autowrite (aw)**

Soll von einer geänderten Datei, z. B. mit dem `:tag` oder dem `:next` Kommando, zu einer anderen geschaltet werden, so gibt *elvis* eine Fehlermeldung aus und wechselt die Datei nicht. Ist die *autowrite* Option gesetzt, so speichert *elvis* die Datei und wechselt zur nächsten.

### **beautify (bf)**

Diese Option löscht, wenn sie gesetzt ist, beim Laden der Datei alle Kontrollzeichen aus dem Text. Wird sie gesetzt, wenn schon eine Datei editiert wird, so ist sie bezüglich der aktuellen



Datei wirkungslos.

### **cc (cc)**

Sie enthält den Namen des C-Compilers, meistens *cc -c* oder *gcc -c*.

### **charattr (ca)**

Viele Textverarbeitungsprogramme erlauben Text unterstrichen, fett oder kursiv darzustellen, indem in den Text Formatkommandos wie `\fU`, `\fB` oder `\fI` eingefügt werden.

Normalerweise behandelt *elvis* diese Kommandos wie normalen Text. Ist die Option *charattr* gesetzt, so interpretiert *elvis* diese Befehle und zeigt den Text in der entsprechenden Darstellungsweise an, wenn das Terminal dies unterstützt und in der Datei */etc/termcap* die richtigen Einträge existieren.

### **columns (co)**

Diese Option enthält die Anzahl der Spalten auf dem Bildschirm.

### **digraph (dig)**

Diese Option steuert, ob Digraphs erkannt werden. Der Standardwert ist *nodigraph*, was bedeutet, daß *elvis* keine Sonderzeichen darstellt.

### **directory (dir)**

*elvis* speichert den zu bearbeitenden Text in Temporärdateien. Diese Option gibt an, in welchem Verzeichnis sie angelegt werden sollen. Diese Option kann nur in der Datei *.exrc* gesetzt werden, da *elvis* nach dem Abarbeiten dieser Datei schon Temporärdateien anlegt, d. h. das Ändern der Option käme zu spät.

### **edcompatible (ed)**

Diese Option beeinflußt das Verhalten des *substitute* Kommandos. Normalerweise ist diese Option ausgeschaltet, was dazu führt, daß alle Optionen des *substitute* Kommandos als nicht gesetzt betrachtet werden, wenn sie nicht explizit angegeben sind. Ist diese Option eingeschaltet, so verwendet *elvis* die Optionen des vorherigen *substitute* Kommandos bis sie explizit geändert werden.

### **equalprg (ep)**

Diese Option enthält den Namen und die Kommandozeilenoptionen des Formatierers, der für das `=` Kommando verwendet werden soll. Die Standardeinstellung ist *fmt*, so daß mit dem `=` Kommando der Text auf 80 Zeichen pro Zeile formatiert wird.

### **errorbells (eb)**

Normalerweise piepst *elvis*, wenn etwas Falsches eingegeben wird. Mit dieser Option wird der Warnton abgeschaltet.

### **exrc**

Diese Option gibt an, ob eine *.exrc* Datei im momentanen Verzeichnis beim Starten von *elvis* ausgeführt werden soll. Wird diese Option in der Datei *.exrc* im Heimatverzeichnis eingeschaltet, so versucht *elvis* die Datei *.exrc* im aktuellen Verzeichnis auszuführen. Diese Option ist hauptsächlich als Schutz gedacht. Sollte z. B. ein böser Mensch auf den Gedanken kommen, mit `echo >/tmp/.exrc '!rm -rf $HOME'` im Verzeichnis */tmp* eine Datei *.exrc* zu erstellen, so wird ein Benutzer, der in */tmp* eine Datei editieren möchte, alle Dateien in seinem Heimatverzeichnis verlieren.

### **exrefresh (er)**

Im `ex mode` gibt *elvis* alle Zeilen einzeln auf dem Bildschirm aus. Wird diese Option gesetzt,

so schreibt *elvis* alle Zeilen auf einmal. Ist z. B. der `write()` Systemaufruf sehr Prozessorzeit intensiv, oder wird eine Fensterumgebung verwendet, so kann es sich anbieten, diese Option auf falsch zu setzen, da einige Fensterumgebungen den Text wesentlich schneller ausgeben, wenn mehrere Zeilen auf einmal geschrieben werden. Diese Option hat keinen Einfluß auf den ``visual command mode'` oder den ``input mode'`.

### **flash (vbell)**

Unterstützt der Termcapeintrag eine optische Alternative zum Warnton, so wird diese Option gesetzt. Ist keine Unterstützung vorhanden, so wird sie von *elvis* auf falsch gesetzt und läßt sich auch nicht einschalten.

### **flipcase (fc)**

Diese Option steuert die Umwandlung von Groß- in Kleinbuchstaben und umgekehrt bei nicht ASCII Zeichen. Die in der Zeichenkette angegebenen Zeichen werden als Paare interpretiert. Wird der `~` Befehl auf einem nicht ASCII Zeichen ausgeführt, so vergleicht *elvis* das Zeichen mit der Liste und ersetzt es durch das andere des Paares.

### **hideformat (hf)**

Viele Textformatierer erwarten im Text Formatkommandos, die mit einem ``.'` beginnen. Normalerweise zeigt *elvis* diese Zeilen wie normalen Text an. Ist die *hideformat* Option gesetzt, so werden diese Zeilen als Leerzeilen angezeigt.

### **ignorecase (ic)**

Normalerweise unterscheidet *elvis* beim Textsuchen zwischen Groß- und Kleinbuchstaben. Mit dieser Option läßt sich diese Unterscheidung abschalten.

### **inputmode (im)**

Diese Option sollte in *.exrc* gesetzt werden und bringt *elvis* dazu im ``input mode'` zu starten. Das Verlassen des ``input modes'` mit `ESC` ist nach wie vor möglich.

### **keytime (kt)**

Auf den meisten Terminals liefern die Cursortasten zusammengesetzte Tastaturcodes. Diese Übertragung benötigt eine bestimmte Zeit, bis die komplette Sequenz eingegangen ist. Die *keytime* Option erlaubt, die maximale Übertragungszeit für die komplette Sequenz anzugeben. Auf den meisten Systemen erfolgt (wie bei Linux) die Angabe in zehntel Sekunden zwischen den einzelnen Zeichen. Die *keytime* Option auf 1 zu setzen, sollte vermieden werden, da viele Systeme nur die Anzahl der CPU-Takte zählen, und so, wenn eine Taste kurz vor Beginn eines neuen Takts gedrückt wird, kaum Zeit zum Einlesen der Sequenz zur Verfügung steht. Hat das System relativ lange Antwortzeiten, oder läuft *elvis* unter einer grafischen Benutzeroberfläche, so sollte bei *keytime* mindestens eine Sekunde angegeben werden. Als Sonderfall kann *keytime* auf 0 gesetzt werden. Dadurch wird das Timeout abgeschaltet, was dazuführen kann, daß *elvis*, wenn die Cursortasten Escape-Sequenzen senden, ewig wartet und nicht in den ``command mode'` zurückkehrt. Diese Option ist eine Erweiterung der Timeout-Option bei *ex/vi*.

### **keywordprg (kp)**

*elvis* hat eine besondere Schlüsselwortfunktion. Befindet sich der Cursor auf einem Wort, so kann der Benutzer `SHIFT-K` eingeben, und *elvis* benutzt ein anderes Programm, um das Schlüsselwort nachzuschlagen und zusätzliche Informationen anzuzeigen. Diese Option gibt an, welches Programm verwendet werden soll. Der Standardwert ist `„ref“`. Dieses Programm schlägt Definitionen von C-Funktionen in einer Datei namens *refs* nach, die von dem Programm *ctags* generiert wurde. Dieser Programmaufruf kann durch einen anderen ersetzt werden, z. B. durch eine Rechtschreibhilfe oder durch ein Online-Manual. *elvis* startet das Programm mit

dem Schlüsselwort als einzigem Kommandozeilenparameter. Das Programm sollte seine Ausgabe auf die Standardausgabe schreiben und als Status 0 zurückliefern.

### **lines (ln)**

Diese Option enthält die Anzahl der Bildschirmzeilen.

### **list (li)**

Im *nolist* Modus zeigt *elvis* den Text ``normal'` auf dem Bildschirm an, d. h. Tabulatorzeichen werden zu einer bestimmten Anzahl Leerzeichen umgewandelt. Wird die *list* Option gesetzt, so zeigt *elvis* für jedes Tabulatorzeichen `^I` und am Zeilenende ein `$` an.

### **magic (ma)**

*elvis'* Suchmechanismus kann reguläre Ausdrücke auswerten. Reguläre Ausdrücke sind Zeichenketten, in denen einige Zeichen eine Sonderbedeutung haben. Normalerweise ist die *magic* Option gesetzt, d. h. *elvis* weist einigen Zeichen Sonderbedeutungen zu. Wird diese Option ausgeschaltet, so verlieren alle Zeichen außer `^` und `$` ihre Sonderbedeutung.

### **make (mak)**

Das `:make` Kommando startet das „*make*“ Dienstprogramm. Die *make* Option enthält den Namen und die Kommandozeilenoptionen des Dienstprogramms.

### **mesg**

Diese Option wird von *elvis* ignoriert.

### **modelines (ml)**

*elvis* unterstützt Moduszeilen. Moduszeilen sind Zeilen am Anfang oder Ende des Textes, die typischerweise Einträge wie `„ex:set ts=5 ca kp=spell wm=15“` enthalten. Anderer Text darf ebenfalls in Moduszeilen vorkommen, so daß sie z. B. als Kommentar geschrieben werden können `/* „ex:set ts=5 ca kp=spell wm=15“ */`. Normalerweise werden die Moduszeilen aus Sicherheitsgründen ignoriert. Diese Option muß in `.exrc` gesetzt werden.

### **more**

Wenn *elvis* im ``visual mode'` mehrere Meldungen in der letzten Bildschirmzeile ausgeben muß, so wartet er mit dem Anzeigen der nächsten bis eine Taste gedrückt wurde. Wird die Option auf falsch gesetzt, so wartet *elvis* nicht. Das bedeutet, daß nur die letzte Meldung gelesen werden kann. Sie ist aber meistens auch die Wichtigste.

### **nearscroll (ns)**

Die Zeile, auf der der Cursor steht, ist immer auf dem Bildschirm. Wird der Cursor zu einer Zeile bewegt, die sich nicht auf dem Bildschirm befindet so scrollt *elvis*, wenn die Zeile nicht weit entfernt ist, oder er baut den Bildschirm neu auf. Die *nearscroll* Option definiert, was von *elvis* als ``weiter entfernt'` verstanden werden soll. Wird die Option z. B. auf 1 gesetzt, so scrollt *elvis* maximal eine Zeile. Setzt man die Option auf 0, so wird das Scrollen abgeschaltet und *elvis* baut den Bildschirm immer neu auf.

### **novice (nov)**

Das Kommando `:set novice` ist synonym zu `:set nomagic report=1 showmode`.

### **number (nu)**

Über die Option *number* wird gesteuert, ob *elvis* vor jeder Zeile eine Zeilennummer ausgibt. Die Zeilennummern sind nicht Teil des Textes. Wird die Datei gespeichert, so wird sie ohne Zeilennummern auf die Festplatte geschrieben.

### **paragraphs (pa)**

Die `{` und `}` Kommandos bewegen den Cursor jeweils um einen Absatz vorwärts oder zurück. Absätze können durch Leerzeilen oder 'Punktcommandos' eines Textformatierers voneinander getrennt werden. Die Option *paragraphs* ermöglicht *elvis* so zu konfigurieren, daß er mit verschiedenen Textformatierern korrekt zusammenarbeitet. *elvis* geht davon aus, daß ein Formatkommando aus einem ``.'` mit einem oder zwei anschließenden Zeichen besteht. Die *paragraphs* Option besteht aus einer Zeichenkette, in der jedes Zeichenpaar eine Befehlskombination des Absatzkommandos des Formatierers darstellt.

#### **prompt (pr)**

Wird die Option auf falsch gesetzt, so gibt *elvis* kein ``:` mehr aus, wenn er die Eingabe eines `ex`-Kommandos erwartet. Diese Option ist nur hilfreich, wenn *elvis* auf einer extrem langsamen Maschine verwendet wird.

#### **readonly (ro)**

Normalerweise erlaubt *elvis* das Schreiben jeglicher Dateien, auf die der Benutzer Schreibberechtigung besitzt. Besitzt der Benutzer keine Schreibberechtigung auf die Datei, so kann die geänderte Datei nur in eine andere geschrieben werden. Wird die *readonly* Option gesetzt, so geht *elvis* davon aus, daß auf keine Datei geschrieben werden darf. Diese Option ist insbesondere hilfreich, wenn *elvis* nur zum Anzeigen von Dateien verwendet werden soll. So wird verhindert, daß eine Datei versehentlich geändert wird. Diese Option ist normalerweise ausgeschaltet, außer *elvis* wird als *view* gestartet.

#### **remap**

Über diese Option wird gesteuert, wie *elvis* sich verhält, wenn in einer `:map` Angabe Textabschnitte vorkommen, die schon als Mapeintrag existieren. Ist die Option eingeschaltet, so werden Mapeinträge innerhalb eines `:map` Kommandos wie Tastendrücke behandelt. So würde z. B. `:map A B` und `:map B C` dazu führen, daß A zu C übersetzt wird, falls die Option eingeschaltet ist.

#### **report (re)**

Beim Editieren können sich Kommandos auf mehrere Zeilen beziehen. Ändert ein Kommando viele Zeilen, so gibt *elvis* eine Meldung aus, wieviele Zeilen geändert wurden. Diese Option steuert, ab wievielen geänderten Zeilen *elvis* eine Meldung ausgibt.

#### **ruler (ru)**

Wird diese Option eingeschaltet, so gibt *elvis* in der letzten Zeile ständig die momentane Cursorposition in Form von *Zeile,Spalte* aus.

#### **scroll (sc)**

Die Kommandos `^U` und `^D` scrollen normalerweise im Text einen halben Bildschirm vorwärts oder zurück. Dies läßt sich mit dieser Option ändern. Sie enthält die Anzahl Zeilen, um die gescrollt werden soll. Wird das Kommando `^U` oder `^D` mit einem *Anzahl* Parameter aufgerufen, so wird diese Option gleich *Anzahl* gesetzt.

#### **sections (se)**

Die `[` und `]` Kommandos bewegen den Cursor jeweils um ein Kapitel vorwärts oder zurück. Kapitel können durch Leerzeilen oder 'Punktcommandos' eines Textformatierers voneinander getrennt werden. Die Option *paragraphs* ermöglicht *elvis* so zu konfigurieren, daß er mit verschiedenen Textformatierern korrekt zusammenarbeitet. *elvis* geht davon aus, daß ein Formatkommando aus einem ``.'` mit einem oder zwei anschließenden Zeichen besteht. Die *section* Option besteht aus einer Zeichenkette, in der jedes Zeichenpaar eine Befehlskombination des Kapitelkommandos des Formatierers darstellt.

## **shell (sh)**

Startet *elvis* eine Shell beispielsweise durch ein `!` oder `:shell` Kommando, so wird das Kommando dieser Option ausgeführt. Der Standardwert ist `„/bin/sh“`, außer die *SHELL* Variable ist gesetzt. In diesem Fall wird das in dieser Variable genannte Kommando als Standardwert angenommen.

## **shiftwidth (sw)**

Mit den Kommandos `<` und `>` können Zeilen um eine bestimmte Anzahl Spalten ein oder ausgerückt werden. Diese Option enthält die Anzahl der Spalten, um die ein- oder ausgerückt werden soll.

## **showmatch (sm)**

Wird die Option *showmatch* gesetzt, so bewegt *elvis* im ``input mode'`, wenn eine Klammer `{ }` `[ ]` `( )` eingegeben wird, den Cursor kurzzeitig zur jeweils zugehörigen anderen Klammer des Paares.

## **showmode (smd)**

Im ``visual mode'` ist es leicht möglich zu vergessen, in welchem Modus sich *elvis* gerade befindet. Wird diese Option eingeschaltet, so zeigt *elvis* in der rechten unteren Ecke des Bildschirms ständig eine Nachricht an, die den momentanen Modus angibt.

## **sidescroll (ss)**

Bei langen Zeilen scrollt *elvis* den Bildschirm seitwärts. (Hierin unterscheidet er sich von *ex/vi*, der lange Zeilen über mehrere Zeilen verteilt darstellt.) Um die Anzahl der Scrolloperationen zu verkleinern, scrollt *elvis* den Bildschirm immer um mehrere Spalten. Diese Option enthält die Anzahl der Spalten, um die *elvis* den Bildschirm bewegt. Diese Option kann umso kleiner eingestellt werden, je schneller der Rechner den Bildschirm scrollen kann.

## **sync (sy)**

Falls das System abstürzt, kann der größte Teil einer geänderten Datei mit Hilfe der Temporärdatei, die *elvis* verwendet, um Änderungen zu speichern, wiederhergestellt werden. Trotzdem werden von dem Betriebssystem nicht alle Änderungen sofort auf die Festplatte geschrieben. Über die *sync* Option läßt sich festlegen, ob *elvis* die Speicherung der Änderungen dem Betriebssystem überläßt, oder ob nach jeder Änderung ein *sync()* aufgerufen wird. Letzteres ist wesentlich langsamer und auf Mehrbenutzersystemen geradezu unfreundlich, da es die gesamte Systemleistung reduziert.

## **tabstop (ts)**

Normalerweise ist ein Tabulatorzeichen 8 Spalten breit. Über diese Option läßt sich ein anderer Wert einstellen.

## **taglength (tl)**

Diese Option bestimmt die Anzahl der signifikanten Zeichen beim Nachschlagen eines Schlüsselwortes. Als Sonderfall kann die Option auf 0 gesetzt werden. Dann müssen alle Zeichen auf das Schlüsselwort zutreffen.

## **term (te)**

Diese Option kann nur gelesen werden und gibt an, welchen Terminaleintrag aus */etc/termcap* *elvis* verwendet.

## **terse (tr)**

*Ex/vi* verwendet diese Option, um festzulegen, ob lange oder kurze Meldungen ausgegeben werden sollen. *elvis* hat nur einen Satz Meldungen, d. h. diese Option ist wirkungslos.

### **timeout (to)**

Das Kommando `:set notimeout` ist gleichbedeutend mit `:set keytime=0`. Das Kommando `:set timeout` ist gleichbedeutend mit `:set keytime=1`. Dadurch wird das Verhalten der ESC-Taste festgelegt.

### **warn (wa)**

Wurde eine Datei geändert, ohne gespeichert zu werden, so gibt *elvis* vor einem `:!Befehl` Kommando eine Warnung aus. Ebenso gibt *elvis* eine Meldung nach einem erfolgreichen Suchen, das das Dateiende überschritten hat, aus. Die *nowarn* Option unterbindet dieses Verhalten.

### **window (wi)**

Diese Option bestimmt, wieviele Zeilen des Bildschirms nach einem langen Bewegungskommando neu aufgebaut werden. Auf schnellen Terminals wird diese Option auf die Anzahl der Bildschirmzeilen minus eins gesetzt. Damit wird der gesamte Bildschirm um den Cursor herum zur Textausgabe genutzt. Auf einem langsamen Terminal bietet es sich an, diese Zahl zu reduzieren, damit beispielsweise ein *n* Kommando nach einem Suchen schneller ausgeführt werden kann.

### **wrapmargin (wm)**

Normalerweise können sehr lange Zeilen eingegeben werden. Wird die Option auf einen anderen Wert als 0 gesetzt, so fügt *elvis* automatisch einen Zeilenumbruch an einem Wortanfang ein, wenn die Zeile zu nah an den rechten Bildschirmrand herankommt. So würde auf einem 80-Zeichen Bildschirm das Kommando `:set wm=10` dazu führen, daß alle Zeilen auf höchstens 70 Zeichen pro Zeile umgebrochen werden.

### **wrapscan (ws)**

Wird nach einer Textstelle gesucht, so findet sie *elvis* unabhängig von der Position innerhalb der Datei. Erreicht *elvis* beim Suchen das Dateiende, so zeigt er in der linken unteren Bildschirmcke die Meldung „*wrapped*“ und setzt das Suchen in der ersten Zeile der Datei fort. Wird die *wrapscan* Option ausgeschaltet, so beendet *elvis* das Suchen am Dateiende.

### **writeany (wr)**

Das Auschalten dieser Option schützt existierende Dateien vor versehentlichem Überschreiben. So würde das Kommando `:w foo` nicht ausgeführt werden, wenn die Datei *foo* schon existiert. Das Kommando `:w! foo` wird unabhängig von der Einstellung der Option immer ausgeführt.

## **Zwischenspeicher (Puffer)**

Wenn *elvis* Text löscht, wird er in einen Puffer kopiert. Dies geschieht sowohl im ``visual mode'`, als auch im ``ex mode'`. Es gibt keine Grenze, wieviel Text in einem Zwischenspeicher gespeichert werden kann. *elvis* besitzt 36 Zwischenpuffer, 26 mit Namen *a-z*, 9 unbenannte *1-9* und einen besonderen ``.'`. Im ``ex mode'` verwenden die Kommandos `:move` und `:copy` einen Puffer, um den Text zwischenzuspeichern.

### **Text zwischenspeichern**

Im ``visual mode'` benutzen die Kommandos *d*, *y*, *c*, *C*, *s* und *x* die Zwischenspeicher *1-9*. Standardmäßig wird der Text in Puffer *1* gespeichert. Der Inhalt des ersten Puffers kommt in den zweiten. Der Inhalt des zweiten in den dritten. Der Inhalt des neunten Puffers geht verloren. Auf diese

Weise bleibt der Text, der in den letzten neun Kommandos gelöscht wurde, erhalten. Ebenso kann Text in einen benannten Zwischenspeicher kopiert werden. Damit der Text in den gewünschten Puffer kopiert wird, muß der Name des Puffers mit einem doppelten Anführungszeichen vor dem Befehl angegeben werden. In diesem Fall werden die Puffer 1-9 nicht verändert. Um Text an einen Puffer anzuhängen, wird sein Name als Großbuchstabe vor dem Kommando angegeben. Der Puffer `.' hat eine Sonderbedeutung. Er enthält die Eingaben des letzten `input modes'. Er wird dazu verwendet, das `.' und ^A Kommando zu implementieren. Um im `ex mode' zusammen mit den Kommandos *:delete*, *:change* und *:yank* Text in einen benannten Puffer zu kopieren, muß der Puffer nach dem Befehl angegeben werden (z. B. *:20,30y a*). Auf die Puffer 2-9 und `.' kann nicht direkt, schreibend zugegriffen werden.

## Einfügen aus einem Puffer

Es gibt zwei Arten Text einzufügen: den Zeilenmodus und den Zeichenmodus. Enthält ein Puffer ganze Zeilen (z. B. von einem *dd* Kommando), so wird der Zeilenmodus verwendet. Enthält ein Puffer nur Teile von Zeilen (z. B. von einem *d3w* Kommando), wird der Zeichenmodus verwendet. Die `ex'-Kommandos schneiden immer ganze Zeilen aus. Im Zeichenmodus wird der Text in die Zeile eingefügt, in der der Cursor steht. Im Zeilenmodus wird der Text in eine neue Zeile oberhalb oder unterhalb der aktuellen Cursorposition eingefügt. Im `visual mode' fügen die Kommandos *p* und *P* Text aus einem Puffer ein. Standardmäßig wird der Puffer 1 verwendet. Soll ein anderer Puffer Verwendung finden, so muß sein Name vor dem Kommando angegeben werden (z. B. *„ap* fügt den Puffer *a* vor dem Cursor ein). Das *p* Kommando fügt den Pufferinhalt vor dem Cursor ein, das *P* Kommando dahinter. Im `ex mode' wird der Text nach einer angegebenen Zeile eingefügt. Soll ein anderer als der Puffer 1 verwendet werden, muß sein Name nach dem Kommando stehen.

## Makros

Der Inhalt eines benannten Puffers kann als Kommandofolge ausgeführt werden. Um die Kommandos in den Puffer zu bringen, werden sie zuerst in die Datei geschrieben und anschließend in einen benannten Puffer gelöscht. Um den Inhalt eines Puffers als `ex' Befehle auszuführen wird das `ex' Kommando *@* verwendet (*:@z* führt den Inhalt des Puffers *z* aus). Soll der Pufferinhalt als `vi' Kommandos ausgeführt werden, wird das `visual mode' Kommando *@* verwendet. Die beiden *@* Kommandos unterscheiden sich dadurch, daß beim `ex'-Kommando der Puffer **nach** dem Befehl angegeben wird, und der Inhalt als `ex' Befehle interpretiert wird. Im `vi' Kommando wird der Puffer **vor** dem Kommando angegeben, und der Pufferinhalt wird als `vi'-Befehle interpretiert. Bei der Verwendung des `vi'-Kommandos *@* ist zu beachten, daß der Pufferinhalt Zeichen für Zeichen interpretiert wird. Jedes Zeichen wird als Tastendruck betrachtet. Ist der Pufferinhalt z. B. mit *„zdd* ausgeschnitten worden, so wird das Zeilenvorschubzeichen am Ende der Zeile ebenfalls als Tastendruck ausgeführt und der Cursor eine Zeile abwärts bewegt. Um dieses Verhalten zu verhindern, sollte der Puffer mit *0,„z D* ausgeschnitten werden. Obwohl in Zwischenspeichern nahezu jede beliebige Textmenge gespeichert werden kann, kann *elvis* nur kleine Puffer als Makros ausführen. Ist ein Puffer zu groß, um ausgeführt werden zu können, gibt *elvis* eine Fehlermeldung aus. Die maximale Makrogröße liegt bei ca. 1000 Zeichen. Desweiteren ist es nicht möglich *:@* Befehle zu verschachteln, sie aus der Datei *.exrc* heraus auszuführen, oder aus einem *:@* Kommando heraus einen *:source* Befehl aufzurufen.

# Wechseln der Arbeitsdatei

Nach dem Starten von *elvis* sind alle Puffer leer. Wird (z. B. mit einem `:n` Befehl) die Datei gewechselt, so werden die 9 anonymen Puffer gelöscht. Der Inhalt der Puffer *a-z* bleibt jedoch erhalten.

## Autor:

Steve Kirkendall

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [elvrec](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [egrep](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)



## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
- 

# elvrec

## Funktion:

**elvrec** Stellt die geänderte Version einer Datei wieder her, falls während des Arbeitens mit `elvis` ein Systemabsturz aufgetreten ist.

## Syntax:

**elvrec** [*Datei* [*neueDatei*]]

## Beschreibung:

Ist während des Editierens mit `elvis` ein Systemabsturz oder Stromausfall aufgetreten, können oft große Teile der letzten Änderungen mit Hilfe der Temporärdateien, die `elvis` während des Editierens anlegt, wiederhergestellt werden.

Beim Neustart des Systems wird die editierte Datei nicht automatisch überschrieben, sondern [elvprsv](#) sichert die noch vorhandenen Änderungen in `/usr/preserve` und erstellt einen Index aller geretteten Dateien..

`elvrec` sucht nach den geretteten Änderungen und aktualisiert die *Datei*, bzw. speichert die aktualisierte Datei unter *neueDatei* ab.

Um eine Liste aller wiederherstellbaren Dateien zu bekommen muß `elvrec` ohne Parameter aufgerufen werden.

## Autor:

Steve Kirkendall

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# env

## Funktion:

**env** (environment) führt einen Befehl in veränderter Umgebung aus

## Syntax:

```
env [-] [-i] [-u Name] [-ignore-environment] [-unset=Name]  
[Name=Wert]... [Befehl [Argumente...]]
```

## Beschreibung:

Mit **env** lassen sich einzelne Unix-Kommandos in veränderter Umgebung ausführen, ohne das Environment der Shell zu verändern. Mit dem Ausdruck *Name=Wert* läßt sich die Umgebungsvariable *Name* ändern bzw. der Umgebung hinzufügen. Der Wert kann leer sein. Eine leere Umgebungsvariable ist verschieden von einer mit `-unset` gelöschtten.

**Achtung!** Die Umgebungsvariablen `PATH`, `IFS`, `PPID`, `PS1`, `PS2`, `UID` und `EUID` können nicht aus der Umgebung entfernt werden (siehe auch beim Shellkommando [unset](#)). Wenn diese Variablen geleert werden, erscheinen in der neuen Prozeßumgebung automatisch die voreingestellten Werte.

## Optionen:

`-u Name`

(unset) entfernt die Umgebungsvariable *Name* aus dem Environment

``-'` oder `-i`

(ignore) löscht alle Umgebungsvariablen

## Autor:

Richard Mlynarik und David MacKenzie



## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# expand

## Funktion:

**expand** ersetzt Tabulatorzeichen durch Folgen von Leerzeichen

## Syntax:

```
expand [-Tab1[,Tab2] [, ...]] [-tTab1[,Tab2[, ...]]] [-i]  
[-tabs=Tab1[,Tab2[, ...]]] [-initial] [Datei...]
```

## Beschreibung:

**expand** liest eine Textdatei und ersetzt alle Tabulatoren durch entsprechende Folgen von Leerzeichen. In der Voreinstellung werden alle Tabulatoren ersetzt und eine Tabellenbreite von 8 Zeichen angenommen. BACKSPACE-Zeichen werden unverändert ausgegeben.

Wenn anstelle eines Dateinamens ein '-' oder gar kein Dateiname angegeben ist, wird von der Standardeingabe gelesen.

## Optionen:

**-*Tab1***

setzt die Tabellenbreite für alle Spalten auf *Tab1* anstelle von 8

**-*Tab1, Tab2, ...***

setzt die erste Tabellenspalte auf *Tab1*, die zweite auf *Tab2* und so weiter; wenn im Text mehr Tabulatoren auftauchen als bei der Option angegeben, werden alle folgenden Tabulatoren durch einzelne Leerzeichen ersetzt

**-t *Tab1* ...**

macht nichts anderes als die oben beschriebenen Optionen

**-i**

konvertiert nur die führenden Tabulatoren jeder Zeile in Leerzeichen

**Autor:**

David MacKenzie

---

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
- 

# expr

## Funktion:

**expr** (expression) bearbeitet einen Ausdruck

## Syntax:

**expr** *Ausdruck* ...

## Beschreibung:

**expr** bewertet oder berechnet einen oder mehrere *Ausdrücke* und gibt das Ergebnis auf die Standardausgabe. Ein Ausdruck besteht aus Zahlen oder Zeichenketten, die durch Operatoren verbunden sind. Eine Zeichenkette braucht nicht in Anführungszeichen eingeschlossen werden. Ob eine Ziffernfolge als Zeichenkette oder als Zahl bearbeitet wird, hängt vom Operator und der Position der Ziffernfolge im Ausdruck ab.

Folgende Operatoren werden erkannt:

|  
*Ausdruck1*|*Ausdruck2* liefert *Ausdruck1*, wenn dieser nicht leer oder gleich 0 ist. Anderenfalls wird *Ausdruck2* ausgegeben.

&

*Ausdruck1*&*Ausdruck2* ist gleich 0, wenn einer der beiden Ausdrücke leer oder 0 ist. Sonst wird *Ausdruck1* ausgegeben.

<, <=, =, !=, >=, >

Vergleicht zwei Ausdrücke und liefert 1, wenn die Relation stimmt, anderenfalls 0. Es wird zuerst versucht, beide Ausdrücke numerisch zu vergleichen. Wenn mindestens einer der Ausdrücke keinen numerischen Wert hat, werden die Ausdrücke lexikografisch verglichen.

+ - \* / %

verknüpft die Ausdrücke arithmetisch. Wenn einer der Ausdrücke keinen numerischen Wert hat, wird eine Fehlermeldung ausgegeben. Der %-Operator liefert den Rest bei ganzzahliger Division (Modulo).

:

**Ausdruck1:Ausdruck2** wendet den regulären *Ausdruck2* auf die Zeichenkette *Ausdruck1* an und liefert die Anzahl der passenden Zeichen oder den, auf den von `(` und `)` eingeschlossenen Teil von *Ausdruck2* passenden Teil von *Ausdruck1*, zurück. Wenn der *Ausdruck2* auf *Ausdruck1* nicht paßt, liefert der Operator 0.

Der Status von `expr` ist

0

wenn der gesamte bewertete Ausdruck weder leer, noch 0 ist

1

wenn der gesamte bewertete Ausdruck leer oder 0 ist

2

wenn ein Fehler aufgetreten ist

## Autor:

Mike Parker

---

Next	Up	Previous	Contents	Index
------	----	----------	----------	-------

**Next:** [fdformat](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [expand](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

Next: [file](#) Up: [Von GNU's, Muscheln und](#) Previous: [expr](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# fdformat

## Funktion:

**fdformat** formatiert eine Diskette (low-level)

## Syntax:

**fdformat** [-n] *Gerätedatei*

## Beschreibung:

**fdformat** formatiert eine Diskette. Das Diskettenformat wird anhand der im Kernel gespeicherten Parameter erzeugt, es wird durch die entsprechende Gerätedatei ausgewählt. Eine Tabelle mit den Namen und Parametern aller im Kernel gespeicherten Formate finden Sie [hier](#).

Mit **fdformat** können nur Formate bis maximal 21 Sektoren/Spur für 3,5 Zoll und 18 Sektoren/Spur für 5,25 Zoll Disketten erzeugt werden. Andere Formate lassen sich mit **superformat** erstellen.

Auf die Diskette werden nur leere Blöcke geschrieben und kein Dateisystem eingerichtet. Dazu stehen die Kommandos **mkfs** **mkxfs** und [mke2fs](#) für Linux-Dateisysteme und [mformat](#) für MS-DOS Dateisysteme zur Verfügung. Die roh formatierte Diskette kann aber auch von **tar** direkt beschrieben werden.

## Optionen:

**-n**

unterdrückt die Verifizierung der formatierten Diskette

## Autor:

Werner Almesberger





**Next:** [find](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [fdformat](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# file

## Funktion:

**file** bestimmt den Dateityp

## Syntax:

```
file [-c ] [-f Namendatei] [-m Magiedatei] Datei ...
```

## Beschreibung:

**file** versucht die Art oder den Typ der angegebenen *Datei* zu bestimmen. Dazu werden drei Tests durchgeführt: ein Dateisystemtest, ein Kennzahlentest und ein Sprachtest. Der erste erfolgreiche Test führt zur Ausgabe des erkannten Dateityps.

Der erkannte Typ enthält normalerweise eines der Schlüsselwörter ``text'` für Dateien, die ohne Schwierigkeiten angezeigt werden können, ``executable'` für Dateien, die ausführbare Programme enthalten und auf dem einen oder anderen Unix-Rechner auch ausgeführt werden können, und ``data'` für alle anderen Dateien, die normalerweise nicht angezeigt werden können. Nur allgemein bekannte Dateiformate wie `core`-Dateien oder `tar` Archive werden ohne diese Schlüsselwörter benannt.

Der Dateisystemtest wird mit Hilfe des `stat(2)` Systemaufrufs durchgeführt. Hier werden leere Dateien ebenso erkannt wie alle Gerätedateien, Sockets, symbolische Links und andere Spezialdateien.

Der Kennzahlentest kann einige Dateien anhand festgelegter Kennzahlen - sogenannter ``magic numbers'` - erkennen, die sich in der Nähe des Dateianfangs an einer festgelegten Stelle befinden. Mit Hilfe solcher Kennzahlen entscheidet beispielsweise das Betriebssystem, ob eine Datei korrekt ausführbar ist oder nicht. Diese Kennzahlen sind in der Datei `/etc/magic` abgespeichert.

Wenn eine Datei als Text erkannt ist, versucht **file** noch, die (Programmier-)Sprache zu erkennen, indem es nach bestimmten Schlüsselwörtern sucht. Auf diese Weise kann beispielsweise C-Quelltext oder die Eingabe für den `groff` Textprozessor erkannt werden.

## Optionen:

–m *Magiedatei*

benutzt die benannte *Magiedatei* anstelle von */etc/magic* für den Kennzahlentest

–C

gibt den interpretierten Inhalt der Kennzahlendatei für Testzwecke aus

–f *Namendatei*

veranlaßt *file*, die Namen der zu untersuchenden Programme aus der *Namendatei* zu lesen; in der *Namendatei* werden die Dateinamen durch Zeilenende getrennt aufgeführt

## Autor:

Ian F. Darwin

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [find](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [fdformat](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Tests:](#)
  - [Aktionen:](#)
  - [Operatoren:](#)
  - [Beispiel:](#)
- 

# find

## Funktion:

**find** sucht nach bestimmten Dateien

## Syntax:

```
find [Verzeichnis] [-Option ...] [-Test ...] [-Aktion ...]
```

## Beschreibung:

**find** durchsucht eine oder mehrere Verzeichnishierarchien nach Dateien mit bestimmten Eigenschaften und führt damit bestimmte Aktionen aus. Die Eigenschaften können durch Tests bestimmt werden.

Optionen, Tests und Aktionen können mit Operatoren zusammengefaßt werden. **find** bewertet für jede Datei in den Verzeichnishierarchien die Optionen, Tests und Aktionen von links nach rechts, bis ein falscher Wahrheitswert auftaucht oder die Kommandozeilenargumente zu Ende sind.

Das erste Argument, das mit einem ``-'`, einer Klammer ``(',')``, einem Komma ``,''` oder einem Ausrufezeichen ``!'`` beginnt, wird als Anfang einer Option oder eines Tests interpretiert. Alle Argumente davor werden als Verzeichnisnamen interpretiert.

Wenn kein Verzeichnis angegeben ist, wird das aktuelle Verzeichnis genommen. Wenn keine Aktion angegeben ist, wird die Aktion ``-print'` ausgeführt.

Der Status von **find** ist Null, wenn alle Aktionen erfolgreich waren, im Fehlerfall ist der Status größer als Null.

## Optionen:

Die Optionen bestimmen das allgemeine Verhalten des Kommandos und beziehen sich nicht auf spezielle Dateien. Die Optionen sind immer wahr.

`-daystart`

mißt die Zeiten für die `-amin`, `-atime`, `-cmin`, `-ctime`, `-mmin` und `-mtime` Eigenschaften vom Beginn des aktuellen Tages anstelle der letzten 24 Stunden

`-depth`

bearbeitet den Inhalt jedes Verzeichnisses vor dem Verzeichnis selbst

`-follow`

folgt den symbolischen Links; diese Option schließt `-noleaf` mit ein

`-maxdepth` **Ebenen**

steigt bis zu der gegebenen Zahl von Ebenen im Verzeichnisbaum auf (in der Hierarchie ab); bei 0 Ebenen werden die Tests nur auf die in der Kommandozeile übergebenen Dateien und Verzeichnisnamen angewendet

`-mindepth` **Ebenen**

steigt mindestens die gegebene Zahl von Ebenen im Verzeichnisbaum auf (in der Hierarchie ab); bei einer Ebene werden die in der Kommandozeile genannten Dateien und Verzeichnisnamen nicht bearbeitet

`-noleaf`

erzwingt die Bearbeitung aller Verzeichniseinträge; normalerweise kann davon ausgegangen werden, daß jedes Linux-Verzeichnis wenigstens zwei (harte) Links enthält: das Verzeichnis ``.'` ist ein Link auf das Verzeichnis selbst, und jedes Unterverzeichnis enthält den Eintrag ``..'` als Link auf das Oberverzeichnis; wenn `find` bei der Untersuchung eines Verzeichnisses zwei Unterverzeichnisse weniger untersucht hat, als das Verzeichnis Links zählt, kann deshalb normalerweise die weitere Suche beendet werden

`-version`

gibt die Versionsnummer auf die Standardfehlerausgabe

`-xdev`

durchsucht keine Verzeichnisse in anderen Dateisystemen (auf anderen Partitionen)

## Tests:

Alle numerischen Argumente können auf drei Arten angegeben werden:

`+N`

steht für alle Zahlen größer als *N*

`-N`

steht für alle Zahlen kleiner als *N*

*N*

steht für genau *N*

Alle Tests werden auf die Dateien in den angegebenen Verzeichnissen einzeln angewendet. Die Tests liefern einen Wahrheitswert von 0 (Wahr), wenn der Test erfolgreich war.

Die Tests auf die erweiterten Zeitmarken (Zugriff und Erstellung) werden nur in solchen Verzeichnissen korrekt behandelt, die auf einem der neuen Linux-Dateisysteme angesiedelt sind (e2fs, xiafs, new minix). Auf den anderen Dateisystemen wird nur das Datum der letzten Änderung zuverlässig getestet. Das Ergebnis der anderen Tests hängt davon ab, ob der letzte Zugriff bzw. die letzte Änderung so kurz zurückliegen, daß die veränderte I-Node noch im Arbeitsspeicher (Cache) ist. Dann können auch für die Dateien der alten Dateisysteme alle drei Zeitmarken unterschieden werden.



`-amin N`

auf die Datei ist vor *N* Minuten zugegriffen worden

`-anewer Referenzdatei`

auf die Datei ist vor weniger Zeit zugegriffen worden, als seit der letzten Veränderung der *Referenzdatei* vergangen ist; im Zusammenhang mit ``-follow'` tritt ``-anewer'` nur in Effekt, wenn ``-follow'` vor ``-anewer'` in der Kommandozeile steht

`-atime N`

auf die Datei ist vor *N*\*24 Stunden zugegriffen worden

`-cmin N`

der Status der Datei wurde vor *N* Minuten geändert 

`-cnower Referenzdatei`

der Status der Datei wurde vor weniger Zeit verändert, als seit der letzten Veränderung der *Referenzdatei* vergangen ist; zusammen mit ``-follow'` tritt ``-cnower'` nur in Effekt, wenn ``-follow'` vor ``-cnower'` in der Kommandozeile steht

`-ctime N`

der Dateistatus wurde vor *N*\*24 Stunden geändert

`-empty`

die reguläre Datei oder das Verzeichnis ist leer

`-false`

ist immer falsch

`-fstype Typ`

die Datei ist in einem Dateisystem vom angegebenen *Typ*; unter anderem werden minix, msdos, ext und proc erkannt

`-gid N`

die Datei gehört der Gruppe mit der Kennzahl *N*

`-group Name`

die Datei gehört der Gruppe ``Name'`

`-inum N`

die Datei belegt die Inode mit der Nummer *N*

`-links N`

die Datei hat *N* (harte) Links

`-lname Muster`

die Datei ist ein symbolischer Link auf eine Datei oder ein Verzeichnis mit einem zum *Muster* passenden Namen

-mmin *N*

der Inhalt der Datei wurde vor *N* Minuten verändert

-mtime *N*

der Inhalt der Datei wurde vor *N*\*24 Stunden verändert

-name ***Muster***

der Name der Datei paßt zu dem *Muster*

-newer ***Referenzdatei***

die Datei ist später verändert worden als die *Referenzdatei*; zusammen mit `-follow` tritt `-newer` nur in Effekt, wenn `-follow` vor `-newer` in der Kommandozeile steht

-nouser

die Datei gehört keinem im System eingetragenen Benutzer

-nogroup

die Datei gehört keiner im System angemeldeten Gruppe

-path ***Muster***

der Pfadname der Datei paßt zum *Muster*

-perm ***Modus***

die Zugriffsrechte auf die Datei entsprechen exakt dem *Modus*; der *Modus* kann als Oktalzahl oder mit den bei [chmod](#) beschriebenen Kennungen beschrieben werden, die Kennungen werden auf Modus `'000'` bezogen

-perm -***Modus***

findet alle Dateien, bei denen mindestens alle Zugriffsrechte vom *Modus* gesetzt sind (bytwweise UND Maske)

-perm +***Modus***

findet alle Dateien, bei denen mindestens ein Bit der Zugriffsrechte mit dem *Modus* übereinstimmt (bitweise UND Maske)

-regex ***Muster***

der Pfadname paßt zu dem regulären Ausdruck *Muster*

size *N*[{c,k}]

die Datei belegt *N* Datenblöcke zu 512 Bytes bzw. *N* Bytes und *N* Kilobytes mit nachgestelltem `'c'` oder `'k'`

-true

ist immer wahr

-type ***C***

die Datei ist vom Typ *C*; folgende Typen werden unterschieden:

b

gepufferte Gerätedatei für ein blockorientiertes Gerät

c

ungepufferte Gerätedatei für ein zeichenorientiertes Gerät

d

Verzeichnis

p  
benannte Pipeline (FiFo)

f  
normale Datei


l  
symbolischer Link

s  
Socket

-uid *N*  
die Kennziffer des Eigentümers ist *N*


-used *N*  
auf die Datei ist *N* Tage nach der letzten Änderung zugegriffen worden

-user *Name*  
die Datei gehört dem Anwender '*Name*'

-xtype *C*  
das gleiche wie '-type' für alle Dateien, die keine symbolischen Links sind; wenn die Datei ein symbolischer Link ist und die Option '-follow' nicht gesetzt ist, wird die Datei, auf die der Link zeigt, auf den Typ *C* geprüft; wenn die Option '-follow' gesetzt ist, ist der Test wahr, wenn *C* = 'l' ist 

## Aktionen:

-exec *Kommando* ;  
führt das *Kommando* aus; die Aktion ist wahr, wenn das Kommando einen Status von Null liefert; alle auf den Kommandonamen folgenden Argumente bis zu einem Semikolon ';' werden als Kommandozeilenargumente für das Kommando interpretiert; das Semikolon kann nicht weggelassen werden, und es muß durch mindestens ein Whitespace von der letzten Option getrennt werden (damit es vor der Shell geschützt ist, muß es zusätzlich Quotiert werden); die Konstruktion '{ }' wird durch den Pfadnamen der Datei ersetzt; die Klammern und das Semikolon müssen in der Kommandozeile für find quotiert werden, damit sie nicht von der Shell bearbeitet werden (siehe auch [bei quotes](#))

-fprint *Ausgabedatei*  
schreibt den Pfadnamen der getesteten Datei in die *Ausgabedatei*; wenn die Ausgabedatei nicht existiert, wird sie erzeugt, sonst wird sie erweitert; die Standardausgabe und die Standardfehlerausgabe werden als '/dev/stdout' und '/dev/stderr' angesprochen 

-fprint0 *Ausgabedatei*  
schreibt den Namen der getesteten Datei in die *Ausgabedatei* und schließt die Ausgabe mit einem Nullbyte ab wie '-print0'

-fprintf *Ausgabedatei Format*  
schreibt den Namen der getesteten Datei in die *Ausgabedatei* und benutzt dabei das *Format* mit Sonderzeichen wie bei '-printf'

-ok *Kommando* ;



wie ``-exec'`, vor der Ausführung des Kommandos wird aber noch eine Bestätigung erwartet; nur eine Eingabe, die mit einem ``Y'` oder einem ``y'` beginnt, führt zur Ausführung des Kommandos

`-print`

gibt den vollständigen Pfadnamen der getesteten Datei auf die Standardausgabe

`-print0`

gibt den Pfadnamen der getesteten Datei, von einem Nullbyte abgeschlossen, auf die Standardausgabe; auf diese Weise können auch Pfadnamen korrekt weiterverarbeitet werden, die ein Zeilenende enthalten

`-printf Format`

gibt für die getestete Datei die Zeichenkette *Format* auf der Standardausgabe aus; Format kann verschiedene Sonderzeichen und Platzhalter enthalten, die von `find` bearbeitet werden:

`\a`

Alarmton

`\b`

Rückschritt

`\c`

Abbruch der Ausgabe

`\f`

Seitenvorschub

`\n`

Zeilenvorschub

`\r`

Wagenrücklauf

`\t`

horizontaler Tabulator

`\v`

vertikaler Tabulator

`\\`

der Backslash selbst

ein Backslash, gefolgt von irgendeinem anderen Zeichen wird als normales Zeichen interpretiert und einfach ausgegeben

`%%`

das Prozentzeichen selbst

`%a`

die Zeit des letzten Zugriffs auf die Datei, in dem Format der `ctime`-Funktion

`%Ak`

die Zeit des letzten Zugriffs auf die Datei, in dem von *k* bestimmte Format; *k* hat dabei das gleiche Format wie der entsprechende Parameter der `strftime`-Funktion in C:

@

Sekunden seit dem 1.1.1970 0 Uhr GMT

H

Stunde (00 bis 23)

I

Stunde (01 bis 12)

k

Stunde (0 bis 23)

l

Stunde (1 bis 12)

M

Minute (00 bis 59)

p

PM oder AM

r

Zeit, 12 Stunden (hh:mm:ss: AM/PM)

S

Sekunden (00 bis 61)

T

Zeit, 24 Stunden (hh:mm:ss)

X

Zeit (H:M:S)

Z

Zeitzone, oder nichts

a

abgekürzter Wochentag

A

ausgeschriebener Wochentag

b

abgekürzter Monatsname

B

ausgeschriebener Monatsname

c

Datum und Zeit

d

Tag im Monat

D

Datum (mm/dd/yy)

h

das gleiche wie `b'

j

der Tag im Jahr

m

die Zahl des Monats

U

die Nummer der Woche, Sonntag als erster Wochentag

w

die Zahl des Wochentags

W

die Nummer der Woche, Montag als erster Wochentag

x

Datum (mm/dd/yy)

y

die letzten beiden Stellen der Jahreszahl

Y

die Jahreszahl

%b

die Dateigröße in 512 Byte Blöcken (aufgerundet)

%c

das Datum der letzten Statusänderung im Format der `Ctime`-Funktion

%Ck

das Datum der letzten Statusänderung im Format der `strftime`-Funktion; Parameter wie oben

%d

die Höhe der Datei im Verzeichnisbaum; Null bedeutet, daß die Datei Kommandozeilenargument ist

%f

der Name der getesteten Datei, ohne Verzeichnisse

%g

der Gruppenname der getesteten Datei oder die Kennzahl, wenn die Gruppe nicht eingetragen ist

%G

die Gruppenkennzahl

%h

die Verzeichnisnamen des Pfadnamen der getesteten Datei

%H

das Kommandozeilenargument (Test), mit dem die Datei gefunden wurde

%i

die Nummer der Inode der getesteten Datei

%k

die aufgerundete Größe der getesteten Datei in Kilobytes

%l

das Objekt, auf die ein symbolischer Link zeigt; leer, wenn die getestete Datei kein symbolischer Link ist

%m

die Zugriffsrechte als Oktalzahl

%n

die Anzahl der harten Links auf die getestete Datei

%p

der Pfadname der Datei

%P

der Pfadname und das Kommandozeilenargument (Test), mit dem die Datei gefunden wurde

%s

die Größe der getesteten Datei in Bytes

%t

die Zeit der letzten Änderung, im `ctime`-Format

%Tk

die Zeit der letzten Änderung, im `strptime`-Format (siehe oben)

%u

der Name des Eigentümers der getesteten Datei oder die Kennzahl, wenn der Benutzer nicht eingetragen ist

%U

die Benutzerkennzahl des Eigentümers der getesteten Datei

-prune

wahr, wenn die Option `-depth` gesetzt ist; sonst falsch

-ls

zeigt die gefundene Datei im Format von `ls -dils` an

## Operatoren:

Die Optionen, Tests und Aktionen können mit Operatoren verknüpft werden. Die Bearbeitung erfolgt prinzipiell von links nach rechts.

**(Ausdruck)**

die Klammern fassen den *Ausdruck* zu einer Operation zusammen

**! Ausdruck**

ist wahr, wenn der *Ausdruck* falsch ist

**-not Ausdruck**

ist ebenfalls wahr, wenn der *Ausdruck* falsch ist

**Ausdruck1 Ausdruck2**

UND Verknüpfung; wenn *Ausdruck1* wahr ist, wird *Ausdruck2* bewertet (ausgeführt)

***Ausdruck1* -a *Ausdruck2***

auch eine UND Verknüpfung

***Ausdruck1* -and *Ausdruck2***

auch eine UND Verknüpfung

***Ausdruck1* -o *Ausdruck2***

ODER Verknüpfung; *Ausdruck2* wird bewertet (ausgeführt), wenn *Ausdruck1* falsch ist

***Ausdruck1* -or *Ausdruck2***

auch eine ODER Verknüpfung

***Ausdruck1* , *Ausdruck2***

Liste; beide Ausdrücke werden immer bewertet (ausgeführt); der Wahrheitswert des gesamten Ausdrucks entspricht dem von *Ausdruck2*

## Beispiel:

Die folgenden Beispiele zeigen typische Anwendungen des `find`-Kommandos aus dem Aufgabenbereich der Systemverwalterin.

Im ersten Beispiel wird das gesamte Dateisystem nach `core`-Files durchsucht. Diese Speicherabzüge bleiben häufig nach Programmabstürzen zurück. Um den dadurch unnütz belegten Festplattenplatz wieder nutzbar zu machen, sollten diese Dateien in regelmäßigen Abständen gelöscht werden. In dem Beispiel werden nur solche Dateien zum Löschen vorgeschlagen, die länger als 5 Tage nicht geöffnet wurden. Auf diese Weise wird verhindert, daß ein möglicherweise gerade zum debuggen eines Programms benutztes `core`-File dem Eigentümer „unter der Hand weg“ gelöscht wird.

```
# find / -atime -5 -name "core*" -ok rm {} \;  
< rm ... /home/she/core > ? n  
< rm ... /usr/src/util-etc-2.1/core > ? y  
< rm ... /usr/openwin/include/images/core_eye.icon > ? n  
< rm ... /usr/openwin/include/images/coredoc.icon > ? n  
# _
```

Im zweiten Beispiel wird das gesamte Dateisystem nach Dateien durchsucht, die mit den Privilegien des Eigentümers ausgeführt werden. Solche Programme sind immer ein potentielltes Sicherheitsrisiko. Deshalb sollte in einem öffentlichen System regelmäßig der Bestand an solchen Dateien durchgesehen und auf Veränderungen geprüft werden.

```
# find / -type f -perm -4000 -exec ls -l {} \;  
-r-sr-sr-x 1 daemon daemon 12328 Aug 12 22:58 /usr/bin/lpq  
-r-sr-sr-x 1 ruth daemon 14048 Aug 12 22:58 /usr/bin/lpr  
-r-sr-xr-x 1 uucp daemon 123908 Mar 24 1993 /usr/bin/cu  
-r-sr-xr-x 1 uucp daemon 87044 Mar 24 1993 /usr/bin/uux  
-r-sr-xr-x 1 uucp daemon 87044 Mar 24 1993 /usr/bin/uucp  
-r-sr-xr-x 1 uucp daemon 37892 Mar 24 1993 /usr/bin/uuname  
-r-sr-xr-x 1 uucp daemon 99332 Mar 24 1993 /usr/bin/uustat  
-rws--x--x 1 ruth ruth 24352 Jan 16 1993 /bin/login
```

-rws--x--x	1	ruth	ruth	16464	Jan	16	1993	/bin/su
-rws--x--x	1	ruth	ruth	16864	Jan	16	1993	/bin/passwd
-rws--x--x	1	ruth	ruth	16292	Jan	16	1993	/bin/gpasswd
-rws--x--x	2	ruth	ruth	11176	Jan	16	1993	/bin/newgrp
-rws--x--x	1	ruth	ruth	9356	Jan	16	1993	/bin/chfn
-rws--x--x	1	ruth	ruth	8232	Jan	16	1993	/bin/chsh
-rws--x--x	1	ruth	ruth	16264	Jan	16	1993	/bin/chage
-rws--x--x	2	ruth	ruth	11176	Jan	16	1993	/bin/sg
---s--x--x	1	ruth	ruth	13316	Aug	13	07:14	/etc/traceroute

## Autor:

Eric Decker, David MacKenzie, Jay Plett und Tim Wood

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [fold](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [file](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

**Next:** [free](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [find](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# fold

## Funktion:

**fold** bricht Zeilen ab einer bestimmten Länge um

## Syntax:

**fold** [-bs] [-w *Länge*] [-bytes] [-spaces] [-width=*Länge*] [*Datei* ...]

## Beschreibung:

**fold** liest die Datei(en) oder die Standardeingabe, wenn keine Datei oder anstelle einer Datei '-' angegeben ist, und schreibt den Inhalt auf die Standardausgabe. Dabei werden Zeilen ab einer bestimmten Länge umgebrochen. Voreingestellte Länge ist 80 Zeichen pro Zeile. Ohne weitere Optionen wird einfach nach dem achtzigsten Zeichen ein Zeilenvorschub eingefügt.

**fold** zählt nicht die Zeichen, sondern die Bildschirmspalten, so daß Tabulatoren korrekt behandelt werden. Ein Backspace erniedrigt die Zeichenzahl entsprechend, und ein Wagenrücklauf (carriage return) setzt die Zahl auf Null zurück.

## Optionen:

-b

es werden nicht die Bildschirmspalten, sondern die Zeichen gezählt; die oben genannten Sonderzeichen erhöhen die Zeichenzahl jeweils um Eins wie alle anderen Zeichen auch

-s

der Zeilenumbruch findet bei dem letzten Leerzeichen der Zeile statt, wenn in der Zeile ein Leerzeichen vorhanden ist

-w *Länge*

setzt die maximale Zeilenlänge auf den angegebenen Wert

## Autor:

David MacKenzie

---

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)



Next: [grep](#) Up: [Von GNU's, Muscheln und](#) Previous: [fold](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# free

## Funktion:

**free** zeigt den freien Speicherplatz im Arbeitsspeicher und auf dem Swapdevice an

## Syntax:

```
free [-msribpc] [-d Blockdevice] [-S Intervall] [Swapdevice]
```

## Beschreibung:

Das **free**-Kommando gehört zu der „ps-Suite“. Wie das **ps**-Programm muß es bestimmte Daten direkt aus dem Kernelspeicher lesen. Dazu gibt es wieder die beiden Möglichkeiten, die bei [ps](#) beschrieben sind. Die **free**-Version, die mit dem Prozeßdateisystem arbeitet, erkennt die hier aufgeführten Optionen nicht. Es zeigt nur den freien Speicherplatz im Arbeitsspeicher und auf dem Swapgerät, wie **free -ms**.

## Optionen:

- m  
zeigt nur den freien Speicherplatz im Arbeitsspeicher
- s  
zeigt nur den freien Speicherplatz im Swapdevice
- i  
zeigt die Auslastung der Inodes im Blockdepot
- b  
zeigt die Auslastung der Datenblöcke im Blockdepot
- d *Gerät*  
zeigt nur die Daten für die gepufferten Blöcke vom Gerät
- f

zeigt die Auslastung der Dateikennzeichner (Filedescriptoren)

-r

zeigt die Daten einer aktuellen Anforderung von Blöcken aus dem Blockdepot

-p

zeigt die Anzahl der Seiten anstelle von Kilobytes (bei den Optionen `-m` und `-s`)

-S *Sekunden*

wiederholt die Anzeige nach der angegebenen Zahl Sekunden

## Autor:

Branko Lankester

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [grep](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [fold](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

**Next:** [groff](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [free](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Beispiel:](#)
- 

# grep

## Funktion:


**grep** (global regular expression print) gibt alle Zeilen aus, in denen ein bestimmter regulärer Ausdruck gefunden wird

## Syntax:

```
grep [-CVbchilnsvwX] [-Anzahl] [-AB Anzahl] [[-e ] Ausdruck | -f  
Datei] [Datei ...]
```

## Beschreibung:

**grep** durchsucht die angegebenen *Dateien* (oder die Standardeingabe) nach einem *Ausdruck* und gibt die entsprechenden Zeilen aus. Der Status von **grep** ist 0, wenn der *Ausdruck* gefunden wurde, und sonst 1.

Als *Ausdruck* akzeptiert **grep** reguläre Ausdrücke mit den folgenden Steuerzeichen: 

**c**

ein einzelner Buchstabe paßt auf sich selbst

**.**

ein Punkt paßt auf jeden Buchstaben außer auf das Zeilenende

**\?**

das dem Operator **\?** vorangehende Muster kann null oder einmal vorkommen

**\***

das dem Operator **\*** vorangehende Muster kann 0 mal oder öfter vorkommen

**\+**

das dem Operator **\+** vorangehende Muster kann einmal oder öfter vorkommen

\|

die durch den Operator '\|' verbundenen Argumente werden **oder** verknüpft

^

(Caret) paßt auf den Zeilenanfang

\$

paßt auf das Zeilenende

\<

paßt auf den Wortanfang 

\>

paßt auf das Wortende 

### [**Buchstaben**]

paßt auf alle *Buchstaben*; dabei können einzelne Buchstaben, aber auch Bereiche in der Form '*von-bis*' angegeben werden; wenn der erste Buchstabe nach '[' ein '^' ist, paßt der Ausdruck auf alle Buchstaben außer den Aufgeführten

\( \)

die Klammern fassen Ausdrücke zusammen; außerdem wird der auf den in Klammern eingeschlossene Teil des Musters passende Text markiert und mit einem folgenden '\N' Ausdruck referenziert (Tag)

\N

referenziert die auf das in der N-ten runden Klammern eingeschlossene Muster passende Zeichenkette.

\

jedes der Sonderzeichen kann, durch einen '\' (Backslash) eingeleitet, sich selbst suchen

\b

paßt auf kein Zeichen, sondern auf den Anfang oder das Ende eines Wortes

\B

symbolisiert den Raum innerhalb eines Wortes

\w

paßt auf alle alphanumerischen Zeichen [A-Za-z0-9]

\W

paßt auf alle nichtalphanumerischen Zeichen [^A-Za-z0-9]

Die Rangfolge der Operatoren ist (von der höchsten zur niedrigsten): `(' , `)` , `?` , `\*` , `+` und `|` . Die anderen Operatoren sind mit den anderen Buchstaben gleichrangig.

## Optionen:

-A *Anzahl*

gibt *Anzahl* Zeilen Kontext **nach** jeder gefundenen Zeile aus

-B *Anzahl*

gibt *Anzahl* Zeilen Kontext **vor** jeder gefundenen Zeile aus

-C

gibt 2 Zeilen Kontext **vor und nach** jeder gefundenen Zeile aus

-*Anzahl*

gibt *Anzahl* Zeilen Kontext **vor und nach** jeder gefundenen Zeile aus

-V

gibt die Versionsnummer auf die Standardfehlerausgabe

-b

gibt die Position jeder gefundenen Stelle mit aus

-c

gibt nur die Gesamtzahl der gefundenen Stellen aus

-e *Ausdruck*

sucht nach *Ausdruck*

-f *Datei*

*Datei* enthält die Ausdrücke, nach denen gesucht werden soll.

-h

unterdrückt die Dateinamen vor jeder Fundstelle

-i

ignoriert Groß- und Kleinschreibung

-l

gibt nur die Dateinamen mit Fundstellen aus

-n

gibt die Zeilennummer zu jeder Fundstelle aus

-s

(silent) keine Ausgabe außer Fehlermeldungen

-v

gibt nur Zeilen aus, die den *Ausdruck* nicht enthalten

-w

gibt nur Zeilen aus, in denen der *Ausdruck* als komplettes Wort vorkommt

-x

gibt nur Zeilen aus, die den *Ausdruck* als ganze Zeile enthalten

## Beispiel:

Beispielsweise können Sie mit dem Kommando

```
$ grep "[Pp]rozess[^eio]" Handbuch.tex
```

werden, erscheinen in der neuen Prozeßumgebung automatisch die  
Die \kommando{free} Version, die mit dem Prozeßdateisystem arbeitet,  
natürlich in der Regel mehr als einen lauffähigen Prozeß. Und der  
dem \kommando{ps} Programm, das mit dem Prozeßdateisystem arbeitet,  
\$ \_

feststellen, ob (noch immer) mehr als die in diesem Beispiel vorgestellten unkorrekten Schreibweisen des Wortes Prozeß in diesem Handbuch vorkommen.

**Autor:**

Mike Haertel, James A. Woods und David Olson

---

Next	Up	Previous	Contents	Index
------	----	----------	----------	-------

**Next:** [groff](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [free](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

Subsections

- [Funktion:](#)
- [Syntax:](#)
- [Beschreibung:](#)
- [Optionen:](#)
- [Umgebung:](#)
- [Dateien:](#)
- [Beispiel:](#)
- [Siehe auch:](#)

# groff

## Funktion:

**groff** ist ein reiner Textfresser. Es ernährt sich von den weitverbreiteten `.1'`, `.n'`, `.an'` und `.ms'` Texten. Es lebt in symbiotischer Gemeinschaft mit `grog`, `gsoelim`, `grops`, `grotty` und einigen verwandten Programmen.

## Syntax:

**groff** [ `-tpeszaivhblCENRVZ`] [ `-w Name`] [ `-W Name`] [ `-H Datei`] [ `-m Makro`] [ `-F Verzeichnis`] [ `-T Format`] [ `-f Familie`] [ `-M Verzeichnis`] [ `-d cs`] [ `-r cn`] [ `-n Nummer`] [ `-o Liste`] [ `-P Argument`] [*Datei ...*]

## Beschreibung:

**groff** ist der Kern eines Textformatiersystems. Normalerweise startet **groff** das **gtroff**-Kommando und leitet die Ausgabe durch einen Postprozessor für ein bestimmtes Ausgabegerät. Folgende Ausgabeformate stehen zur Verfügung:

- ps
- Postscript
- dvi
- (DeVice Independent) das TeX Ausgabeformat
- X75
- für X11 Ausgabe mit 75dpi
- X100

für X11 Ausgabe mit 100dpi

ascii

für einfache Druckerausgabe

latin1

für Druckerausgabe mit ISO Latin-1 Zeichensatz

Das Standardformat ist ascii.

Im groff-System stehen außerdem die Präprozessoren `gpic`, `geqn`, `gtbl`, `grefer` und `gsoelim` zur Verfügung.

## Optionen:

-h

gibt einen Hilfstext aus

-e

leitet die Eingabe durch den `geqn`-Präprozessor

-t

leitet die Eingabe durch den `gtbl`-Präprozessor

-p

leitet die Eingabe durch den `gpic`-Präprozessor

-s

leitet die Eingabe durch den `gsoelim`-Präprozessor

-R

leitet die Eingabe durch den `grefer`-Präprozessor; die Übergabe von Kommandozeilenargumenten an `grefer` wird nicht unterstützt

-v

die von `groff` aufgerufenen Programme geben ihre Versionsnummer aus

-V

gibt die von `groff` zusammengestellte Kommandozeile (Pipeline) auf der Standardausgabe aus, anstatt sie auszuführen

-z

unterdrückt die Ausgabe von `gtroff`; nur Fehlermeldungen werden ausgegeben

-Z

unterdrückt den Postprozessor

-P *Argument*

gibt das *Argument* an den Postprozessor weiter; jedes Argument sollte einzeln übergeben werden; den Argumenten wird kein ``-'` vorangestellt

-L *Argument*

gibt das *Argument* an den Spooler weiter

-T *Format*

benutzt Ausgabe*Format*; Voreinstellung ist `ascii`



- N  
übergibt die -N Option an geqn
- a  
produziert reinen ascii-Code, ohne Steuerzeichen
- b  
gibt zusätzliche Information bei Fehlermeldungen
- i  
liest aus der Standardeingabe, nachdem alle Eingabedateien bearbeitet sind
- C  
schaltet in den Kompatibilitätsmodus (zu troff)
- E  
unterdrückt alle Fehlermeldungen
- w **Name**  
erlaubt Warnung *Name*
- W **Name**  
unterdrückt Warnung *Name*
- m **Makro**  
die Datei ``tmac.Makro'` wird gelesen und die darin definierten Makros zum Formatieren des Dokuments benutzt
- o **Liste**  
gibt nur die Seiten aus *Liste* aus; die *Liste* ist eine durch Kommata getrennte Liste von Seitenbereichen
- d **cs**  
definiert das Register *c* mit *s*; dabei ist *c* ein Buchstabe und *s* eine Zeichenkette
- r **cn**  
setzt Register *c* auf *n*. Dabei ist *c* ein Buchstabe und *n* ein numerischer Ausdruck.
- F **Verzeichnis**  
sucht im *Verzeichnis* nach den Fonts
- M **Verzeichnis**  
sucht im *Verzeichnis* nach den Makros
- H **Datei**  
benutzt *Datei* nach der Trennmusterdatei
- f **Familie**  
benutzt *Familie* als Fontfamilie
- n **Nummer**  
setzt die Nummer der ersten Ausgabeseite auf *Nummer*

## Umgebung:

Folgende Umgebungsvariablen werden unterstützt:

### GROFF\_TMAC\_PATH

eine durch Doppelpunkte getrennte Liste von Verzeichnissen, in denen nach Makrodateien gesucht wird

### GROFF\_TYPESETTER

das Standardausgabeformat

### GROFF\_FONT\_PATH

eine durch Doppelpunkte getrennte Liste von Verzeichnissen, in denen nach den Gerätetreibern und den Zeichensätzen für das Ausgabeformat gesucht wird

### GROFF\_HYPHEN

eine Datei mit Mustern für die automatische Trennung durch `groff`

### GROFF\_TMPDIR

ein Verzeichnis, in dem die temporären Dateien von `groff` angelegt werden; wenn kein Verzeichnis angegeben ist, wird das Verzeichnis `/tmp` benutzt

## Dateien:

Die folgenden Dateien werden vom `groff`-System benutzt:

`/usr/lib/groff/hyphen`

die Standardtrennmusterdatei

`/usr/lib/groff/tmac/tmac.Name`

die Makrodatei für ``-mName'`

`/usr/lib/groff/font/devName/DESC`

der Gerätetreiber für das Gerät *Name*

`/usr/lib/groff/font/devName/F`

die Fontdatei für Font ``F'` von Gerät *Name*

`/usr/lib/groff/font/devName/eqnchar`

die Definitionen von `(g)eqn` für das Gerät *Name*

## Beispiel:

In den meisten Programmpaketen sind unformatierte Manualpages enthalten. Wenn Sie sich einen solchen Hilfstext ansehen wollen, ohne ihn gleich in einem Verzeichnis im `MANPATH` zu installieren, können Sie sich die Manualpage „von Hand“ formatieren und mit einem Pager anzeigen lassen:

```
$ groff -mandoc -Tascii foobar.1 > cat.foobar.1
$ less cat.foobar.1
```

## Siehe auch:

[grog\(1\)](#), [gtroff\(1\)](#), [gtbl\(1\)](#), [gpic\(1\)](#), [geqn\(1\)](#), [gsoelim\(1\)](#), [grefer\(1\)](#), [grops\(1\)](#),  
[grodvi\(1\)](#), [grotty\(1\)](#), [groff\\_font\(5\)](#), [groff\\_out\(5\)](#), [groff\\_msr\(7\)](#), [groff\\_me\(7\)](#)

## Autor:

James Clark

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [groups](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [grep](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

Next: [gzip](#) Up: [Von GNU's, Muscheln und](#) Previous: [groff](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Siehe auch:](#)
- 

# groups

## Funktion:

**groups** zeigt alle Gruppen, denen der Benutzer angehört

## Syntax:

**groups** [*Benutzer* ...]

## Beschreibung:

Mit dem **groups**-Kommando können Sie sich eine Liste aller Benutzergruppen ausgeben lassen, denen Sie fest angehören. Mit dem **newgrp**-Kommando können Sie jede dieser Gruppen zu Ihrer aktuell aktiven Benutzergruppe machen.

## Siehe auch:

**groups** ist ein Shellscript und benutzt **id**

**Next:** [head](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [groups](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- 

# gzip

## Funktion:

**gzip** komprimiert Dateien

## Syntax:

**gzip** [-cdfhlLnNqrtvV19] [-S *Endung* [*Datei* ...]

## Beschreibung:

**gzip** komprimiert Dateien mit dem LZ77 Lempel-Ziv Algorithmus. **gzip** erzielt erheblich bessere Kompressionsraten als das mit dem LZW-Algorithmus arbeitende **compress**-Programm. Weil es sich ansonsten sehr ähnlich verhält, ist abzusehen, daß es **compress** als Standardpacker im Bereich der freien Software verdrängen wird. Mit **gzip** können auch Dateien ausgepackt werden, die mit **compress** oder **pack** gepackt wurden. Archive, die mit **zip** gepackt wurden, können mit **gzip** nur ausgepackt werden, wenn sie eine einzige Datei enthalten und mit der „*deflation*“ Methode gepackt wurden.

**gzip** ist kein Archivpacker wie **lharc**, **arj** oder **pkzip**.

**gzip** komprimiert einzelne Dateien, unabhängig davon, ob die resultierende Datei tatsächlich kleiner ist, und ersetzt die Urdatei durch die komprimierte, indem es an den Dateinamen die Endung `.gz` anhängt. Wenn der Dateiname durch Anhängen der Endung unzulässig lang würde, verkürzt **gzip** automatisch den Namen um die erforderliche Anzahl Zeichen.

Die Zeitmarke der Datei und die Zugriffsrechte bleiben beim Komprimieren erhalten. Um die Wiederherstellung der Zeitmarke und des Dateinamens sicherzustellen, werden diese Daten mit eingepackt und können beim entkomprimieren verwendet werden. Außerdem wird eine CRC-Checksumme mit eingepackt, mit der beim Auspacken automatisch die Integrität der Daten geprüft wird.

Wenn `gzip` ohne Dateinamen aufgerufen wird, liest es von der Standardeingabe und schreibt auf die Standardausgabe. Der gleiche Effekt wird erzielt, wenn anstelle einer Eingabedatei ein Minuszeichen '-' angegeben wird.

Wie bei `compress` kann auch `gzip` auf andere Namen gelinkt werden, um bestimmte Aufgaben zu erfüllen.

Unter dem Namen **gunzip** arbeitet es wie `gzip -d`, packt also komprimierte Dateien der oben aufgeführten Formate aus. `gunzip` erwartet die Endung `.gz`, `-gz`, `.tgz`, `.taz`, `.z`, `-z`, `-z` oder `.Z` an dem Dateinamen. Außerdem wird die Datei auf eine „magische Zahl“ überprüft, die mit `gzip` komprimierte Dateien identifiziert. Nach dem Auspacken bleiben die Zugriffsrechte und das Erstellungsdatum der Datei erhalten.

**zcat** arbeitet wie `gzip -dc`, schreibt also die entkomprimierte Datei auf die Standardausgabe und läßt die komprimierte Datei unberührt. Wenn die Eingabedatei die korrekte magische Zahl enthält, wird sie ausgepackt, egal welche Endung der Dateiname hat.

## Optionen:

- c  
schreibt die (ent)komprimierte *Datei* auf die Standardausgabe, anstatt die Datei zu ersetzen
- d  
(decompress) dekomprimiert die *Datei*
- f  
(force) ersetzt bestehende Dateien mit Endung `.gz`; normalerweise fragt `gzip` vor dem Überschreiben solcher Dateien nach
- h  
(help) gibt eine Kurzhilfe zum Programm aus
- l  
(list) zeigt den in einer mit `gzip` komprimierten Datei gespeicherten originalen Dateiname, sowie die originale und die gepackte Größe an; wenn die `-v`-Option gesetzt ist, wird zusätzlich die Zeitmarke und die Checksumme ausgegeben
- L  
(license) gibt eine Kurzfassung des Lizenztextes aus
- n  
(noname) unterdrückt beim Einpacken das Speichern des Dateinamen und der Zeitmarke (nur wenn der Name nicht gekürzt werden muß); beim Auspacken wird die Wiederherstellung des originalen Namens mit der Zeitmarke unterdrückt; diese Option ist Voreinstellung zum Entpacken
- N  
(name) veranlaßt beim Einpacken die Sicherung des originalen Namen und der Zeitmarke in der gepackten Datei und beim Auspacken die Wiederherstellung dieser Daten an der dekomprimierten Datei; diese Option ist Voreinstellung beim Einpacken
- q  
(quiet) unterdrückt alle Warnungen

-r

(recursive) packt alle Dateien in den angegebenen Unterverzeichnissen

-S *Endung*

veranlaßt die Verwendung der neuen *Endung* anstelle von ``.gz'`

-t

(test) prüft die Integrität der angegebenen *Datei*

-v

(verbose) gibt den Namen und den Kompressionsfaktor für jede *Datei* aus

-V

(Version) gibt die Versionsnummer des Programms aus

-*Ziffer*

bestimmt mit einer Ziffer von 1 bis 9 die Kompressionstiefe; 1 bedeutet schnell und schlecht komprimiert, 9 bedeutet langsam und optimal komprimiert

## Siehe auch:

compress(1) und [tar](#)

### Autor:

Jean-Loup Gailly

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [head](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [groups](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

**Next:** [hexdump](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [gzip](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# head

## Funktion:

**head** schreibt den Anfang einer Datei auf die Standardausgabe

## Syntax:

```
head [-c Anzahl[bkm]] [-n Anzahl] [-qv] [-bytes=Anzahl[bkm]]  
[-lines=Anzahl] [-quiet] [-silent] [-verbose] [Datei ...]
```

```
head [-nr bcklmqv] [Datei ...]
```

## Beschreibung:

**head** schreibt die ersten (10) Zeilen von der *Datei* auf den Bildschirm. Wenn keine *Datei* oder `-` angegeben wird, liest **head** von der Standardeingabe. Wird mehr als eine *Datei* angegeben, so wird der Dateiname, in `'>'` und `'<'` eingeschlossen, der Ausgabe vorangestellt.

## Optionen:

**-c *Anzahl***

gibt die angegebene *Anzahl* Bytes aus. Optional kann die Blockgröße durch einen der Zahl folgenden Buchstaben verändert werden:

b

Blocks zu 512 Byte

k

Blocks zu 1 Kilobyte

m

Blocks zu 1 Megabyte

**-n *Anzahl***

gibt die ersten *Anzahl* Zeilen aus



- q  
(quiet) unterdrückt die Ausgabe des Dateinamen
- v  
(verbose) gibt die Dateinamen immer aus
- Anzahl*  
gibt die angegebene Zahl Zeilen aus

**Autor:**

David MacKenzie

---

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- 

# hexdump

## Funktion:

**hexdump** zeigt Dateien in hexadezimalen, dezimalen, oktalen oder ascii Zeichenformat an

## Syntax:

**hexdump** [-bcdovx] [-e *Formatstring*] [-f *Formatdatei*] [-n *Anzahl*] [-s *Anzahl*] *Datei* ...

## Beschreibung:

**hexdump** liest aus der Standardeingabe oder aus einer Datei und gibt die einzelnen Bytes formatiert und kodiert auf die Standardausgabe.

## Optionen:

-b

die Ausgabe erfolgt in sechzehn Spalten zu je einem Byte in oktaler Darstellung mit hexadezimaler Dateiposition

-c

die Ausgabe erfolgt in sechzehn Spalten zu je einem Byte mit hexadezimaler Dateiposition. Druckbare Zeichen werden als solche angezeigt, nichtdruckbare Zeichen werden oktal dargestellt.

-d

die Ausgabe erfolgt in fünf Spalten zu je zwei Byte in dezimaler Darstellung mit hexadezimaler Dateiposition

-e *Formatstring*

legt das Ausgabeformat mit *Formatstring* fest

–f **Formatdatei**

liest die *Formatstrings* aus der *Formatdatei*. Jede Zeile enthält einen *Formatstring*, leere Zeilen und Zeilen deren erstes Zeichen ein '#' ist werden ignoriert.

–n **Anzahl**

gibt nur *Anzahl* Bytes aus

–o

die Ausgabe erfolgt in sechs Spalten zu je zwei Byte in oktaler Darstellung mit hexadezimaler Dateiposition

–s **Anzahl**

überspringt *Anzahl* Bytes am Anfang der Eingabe. *Anzahl* ist eine dezimale Zahl. Beginnt *Anzahl* mit 0x oder 0X wird sie hexadezimal interpretiert, beginnt sie mit 0 wird sie oktal interpretiert. Außerdem kann der Zahl einer der Buchstaben b, k oder m nachgestellt werden. Dadurch wird der Offset zu einem vielfachen von 512 (block) 1024 (kilo) oder 1048576 (mega).

–v

zeigt alle Eingabezeilen an. Normalerweise werden identische Zeilen durch ein einzelnes Asterisk (\*) ersetzt.

–x

die Ausgabe erfolgt in acht Spalten zu je zwei Byte in hexadezimaler Darstellung mit hexadezimaler Dateiposition

## Siehe auch:

[od](#)

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [hostname](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [head](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

**Next:** [id](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [hexdump](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Siehe auch:](#)
- 

# hostname

## Funktion:

**hostname** zeigt oder setzt den (DNS Netzwerk-) Namen des Systems

## Syntax:

**hostname** [-dfshv] [-F *Datei*] [*Name*]

## Beschreibung:

Das **hostname**-Kommando wird benutzt, um den lokalen Rechnernamen anzuzeigen bzw. um ihn zu setzen oder zu ändern.

Ein Rechnername hat eigentlich erst im Netzwerkbetrieb seine wirkliche Bedeutung, trotzdem wird er auch bei allen alleinstehenden Systemen gesetzt und wie im Netzwerk verwaltet. Im Netz besteht ein vollständiger Rechnername (Fully Qualified Domain Name) aus einem Eigennamen und einem

Domainnamen.  Der (DNS-) Domainname bezeichnet das lokale Netz an dem der Rechner hängt.

Um den Namen des Rechners zu ändern, sind Rootprivilegien erforderlich. Normalerweise wird der Rechnername beim Systemstart durch eines der Systeminitialisierungsprogramme gesetzt.

Wenn die Funktionalität von **hostname** auf dem alleinstehenden Rechner ohne Netzwerk vollständig genutzt werden soll, muß der vollständige Rechnername in der Datei `/etc/hosts` als erster Namenseintrag für das Loopback-Device eingetragen sein.

-s

gibt nur den kurzen Eigennamen des Rechners aus

-f

gibt den vollständigen Rechnernamen aus

-d

gibt nur den (DNS-) Domainnamen des lokalen Netzes aus

-h

gibt eine Kurzhilfe zum Programm aus

-v

zeigt die Versionsnummer des Programms

-f *Datei*

bestimmt eine *Datei* aus der der Rechnername gelesen werden soll

## Siehe auch:

[uname](#)

## Autor:

Peter Tobias

---

Next	Up	Previous	Contents	Index
------	----	----------	----------	-------

**Next:** [id](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [hexdump](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# id

## Funktion:

**id** gibt die reale und die effektive User-ID und Gruppen-ID aus

## Syntax:

**id** [-gnruG] [-group] [-name] [-real] [-user] [-groups] [*Username*]

## Beschreibung:

Das **id**-Kommando zeigt die reale Benutzerkennzahl (UID) mit dem Benutzernamen und alle Gruppen, in denen der Anwender eingetragen ist, mit ihren Kennzahlen (GID) und Namen an. Wenn die effektive Benutzerkennung nicht der realen entspricht, wird die effektive Benutzerkennung ebenfalls angezeigt.

## Optionen:

- g  
gibt nur die Gruppen-ID aus
- n  
gibt den User- bzw. den Gruppennamen anstelle der ID (nur in Verbindung mit -u, -g oder -G)
- r  
gibt die reale anstelle der effektiven User- bzw. Gruppen-ID aus (nur in Verbindung mit u, -g oder -G)
- u  
gibt nur die User-ID aus
- G  
gibt die ID's aller Gruppen von User aus

**Autor:**

Arnold Robbins und David MacKenzie

---

*Das Linux Anwenderhandbuch*

(C) 1997 [\*LunetIX\*](#)

**Next:** [join](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [id](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# install

## Funktion:

**install** kopiert Dateien, richtet Verzeichnisse ein und ändert die Zugriffsrechte

## Syntax:

```
install [Optionen] [-s] [-strip] Quelldatei Zieldatei  
install [Optionen] [-s] [-strip] Quelldatei ...Verzeichnis  
install [Optionen] [-d] [-directory] Verzeichnis ...
```

## Beschreibung:

**install** wird normalerweise direkt vom make Utility aufgerufen, um ein neu übersetztes Programm mit allen Hilfs- und Konfigurationsdateien in die dafür vorgesehenen Verzeichnisse zu kopieren, und mit den nötigen Rechten auszustatten. Dabei können auch neue Verzeichnisse angelegt werden.

## Optionen:

-c

ohne Funktion; zur Kompatibilität mit alten Unix Versionen von **install**

-d

erzeugt alle Verzeichnisse und die davor liegenden Verzeichnisse, wenn sie nicht existieren; wenn Modus, Eigentümer und Gruppe bestimmt sind, werden die entsprechenden Werte für alle Verzeichnisse benutzt, sonst gelten die Standardwerte (EUID, EGID und umask)

-g *Gruppe*

ändert die Gruppenkennung in den angegebenen Wert; es werden sowohl der Gruppenname, als auch die Gruppenkennzahl verarbeitet

-m *Modus*

setzt die Zugriffsrechte auf den angegebenen Modus; es werden die bei [chmod](#) erklärten



Oktalzahlen oder Buchstabencodes verarbeitet; die Buchstabencodes basieren auf dem Modus 000; der Standardmodus ist 0755

–o ***Eigentümer***

ändert den Eigentümer der Datei; diese Option steht nur der Systemverwalterin zur Verfügung; es werden sowohl Namen als auch Benutzerkennzahlen verarbeitet

–s

entfernt die Symboltabellen aus den Programmdateien

**Autor:**

David MacKenzie

---

Next	Up	Previous	Contents	Index
------	----	----------	----------	-------

**Next:** [join](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [id](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

Next: [kill](#) Up: [Von GNU's, Muscheln und](#) Previous: [install](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# join

## Funktion:

**join** verknüpft zwei Dateien nach Schlüsselfeldern

## Syntax:

```
join [-a 1|2] [-v 1|2] [-e Zeichenkette] [-o Feldliste ...] [-t  
Buchstabe] [-j[1|2] Feldnr] [-1 Feldnr] [-2 Feldnr] Datei1 Datei2
```

## Beschreibung:

**join** verknüpft zwei (alphabetisch) sortierte Dateien, indem je zwei Zeilen mit identischen Schlüsselfeldern zu einer Ausgabezeile verbunden werden.

Die Schlüsselfelder sind durch Leerzeichen voneinander getrennt. Führende Leerzeichen werden ignoriert. Wenn nicht anders angegeben, ist das erste Feld einer jeden Zeile Schlüsselfeld. Die Ausgabefelder sind ebenfalls durch Leerzeichen voneinander getrennt. Die Ausgabe besteht aus dem Schlüsselfeld, gefolgt von den übrigen Feldern der Datei1, und schließlich aller Felder der passenden Zeilen von Datei2 ohne das Schlüsselfeld.

## Optionen:

**-a *Dateinummer***

fügt in die Ausgabe eine Leerzeile ein, wenn eine Zeile aus *Dateinummer* (1 oder 2) kein Gegenstück hat.

**-e *Zeichenkette***

ersetzt fehlende Eingabefelder in der Ausgabe durch die *Zeichenkette*

**-1 *Feldnr***

benutzt in Datei1 *Feldnr* als Schlüsselfeld

**-2 *Feldnr***

benutzt in Datei2 *Feldnr* als Schlüsselfeld

–j ***Feldnr***

benutzt *Feldnr* als Schlüsselfeld

–o ***Feldliste***

stellt die Ausgabezeilen anhand der *Feldliste* zusammen. Ein Eintrag in der *Feldliste* besteht aus einer *Dateinummer*, einem Punkt und einer *Feldnr*. Beliebige viele solcher Paare *Dateinummer.Feldnr* können, durch Komma oder Leerzeichen getrennt, in der *Feldliste* stehen.

–t ***Buchstabe***

verwendet *Buchstabe* als Feldtrenner

–v ***Dateinummer***

gibt nur die Zeilen aus *Dateinummer* aus, die kein Gegenstück haben.

## Autor:

Mike Haertel

---

Next	Up	Previous	Contents	Index
------	----	----------	----------	-------

**Next:** [kill](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [install](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

**Next:** [less](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [join](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# kill

## Funktion:

**kill** beendet einen Prozeß

## Syntax:

```
kill [-s Signal] [-p] [-a] [-l [Signalnummer]] Prozeß ...
```

## Beschreibung:

**kill** wird benutzt, um außer Kontrolle geratene („aufgehängte“) Prozesse, die sich nicht mehr auf normale Art beenden lassen, zu terminieren (beenden) oder um Signale an bestimmte Prozesse zu senden.

Das *Signal* kann mit seinem Namen oder mit der Singalnummer angegeben werden. Wenn kein Signal spezifiziert ist, wird SIGTERM (15) gesendet.

Ein *Prozeß* kann durch seine Prozeßnummer oder durch seinen Namen bezeichnet werden.

Signale können ohne Rootprivilegien nur an die eigenen Prozesse gesendet werden.

Wenn ein Prozeß ein Signal empfängt können verschiedene Fälle eintreten:

1.

Indem das Anwendungsprogramm eine spezielle Funktion zur Behandlung eines Signals bereitstellt kann dieses Signal abgefangen werden. Signale können damit zur asynchronen Fehler- bzw. Ausnahmebehandlung sowie zu einer primitiven Prozeßkommunikation genutzt werden.

2.

Das Signal kann ignoriert werden. Das ist der Regelfall für alle Signale die nicht abgefangen werden.

3.

Die Signale SIGKILL und SIGSTOP können nicht abgefangen werden, sie werden direkt vom

Kernel behandelt. Mit SIGKILL (9) wird der Prozeß sofort beendet.

In der `bash` ist ein `kill`-Kommando eingebaut, daß dieses externe Programm verdeckt, wenn nicht ausdrücklich mit dem `command-Shell`-Kommando das externe Programm aufgerufen wird. (Siehe beim Shellkommando [kill](#))

## Optionen:

-s *Signal*

sendet das *Signal* anstelle von SIGTERM (15)

-p

gibt die Prozeßnummern der Prozesse aus, an die ein Signal gesendet würde, ohne es zu senden

-a

veranlaßt `kill` auch Prozesse anderer Benutzer einzubeziehen

-l

gibt eine Namensliste aller Signale aus; eine Signalnummer als Argument wird in den entsprechenden Signalnamen übersetzt

## Autor:

Portiert von BSD 4.4 mit Erweiterung von Salvatore Valente

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [less](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [join](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

**Next:** [ln](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [kill](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Kommandos:](#)
  - [Optionen:](#)
  - [Umgebungsvariable](#)
  - [Siehe auch:](#)
- 

# less

## Funktion:

**less** ist ein Pager wie **more**

## Syntax:

```
less [-[+]aBcCdeEfHimMnNqQrsSuUw] [-b Puffer] [-h Zeilen] [-j Zeile]
[-k Schlüsseldatei] [-{oO} Datei] [-p Muster] [-P Prompt] [-t Tag]
[-T Tagdatei] [-x Tab] [-y Zeilen] [-[z] Zeilen] [+Lesskommando]
[Dateiname] ...
```

## Beschreibung:

**less** gibt eine (oder mehrere) Datei(en) seitenweise auf dem Bildschirm aus. Im Unterschied zu **more** erlaubt **less** auch das Zurückblättern in Texten, die aus einer Pipeline gelesen werden. Die voreingestellten Tastaturkommandos sind an die von **more** und **elvis** angelehnt, sie lassen sich aber vom Anwender mit Hilfe des **lesskey**-Kommandos beliebig neu definieren.

## Kommandos:

Die folgenden Tastaturkommandos zur Steuerung der Bildschirmausgabe sind definiert. Alle Kommandos können von einer ganzen Zahl *N* eingeleitet werden, die die Anzahl der Wiederholungen dieses Kommandos angibt.

**h** | **H**

gibt einen Hilfstext aus

LEERZEICHEN | CTRL-V | **f** | CTRL-F

blättert eine Bildschirmseite vorwärts; wenn eine Zahl vorangestellt ist, werden diese Anzahl Zeilen weitergeblättert

z

wie Leerzeichen; wenn eine Zahl vorangestellt ist, wird diese Zahl zur neuen Seitenlänge auch für die weiteren Seiten

RETURN | CTRL-N | CTRL-E | CTRL-J

scrollt den Bildschirm eine (oder die gegebene Anzahl) Zeilen weiter

d | CTRL-D

blättert einen halben Bildschirm (oder die gegebene Anzahl Zeilen) weiter; wenn eine Zahl angegeben ist, wird sie zur Standardweite für alle folgenden CTRL-D und CTRL-U-Kommandos

b | CTRL-V

blättert einen Bildschirm oder die gegebene Anzahl Zeilen zeilenurück

w

wie ESC-V; die gegebene Zahl wird aber der neue Standardwert für das Zurückblättern

y | CTRL-Y | CTRL-P | k | CTRL-K

scrollt den Bildschirm um eine (oder die gegebene Anzahl) Zeile(n) zurück

u | CTRL-U

blättert einen halben Bildschirm (oder die gegebene Anzahl Zeilen) zurück

r | CTRL-R | CTRL-L

schreibt den aktuellen Bildschirm neu

R

schreibt den Bildschirm neu; dabei wird die gesamte gepufferte Eingabe verworfen

F

scrollt den Bildschirm vorwärts; dabei wird versucht weiterzulesen, auch wenn das Dateiende bereits erreicht worden ist

g | < | CTRL-<

geht zur ersten Zeile oder der Zeile mit der entsprechenden Zahl

G | > | CTRL->

geht zur letzten Zeile oder zu der Zeile mit der entsprechenden Zahl (vom Dateiende gezählt)

p | %

macht nur Sinn, wenn es von einer Zahl zwischen 0 und 100 eingeleitet wird, und geht dann zu der dieser Prozentzahl entsprechenden Stelle; wenn von der Standardeingabe gelesen wird, kann dieses Kommando erst benutzt werden, nachdem die Eingabe abgeschlossen ist

}

wenn eine geschweifte Klammer auf der ersten Bildschirmzeile steht, wird die Anzeige so gescrollt, daß die dazugehörige schließende Klammer auf der letzten Bildschirmzeile steht; wenn mehr als eine geschweifte Klammer auf der ersten Zeile geöffnet wird, kann mit dem Zahlenargument bestimmt werden, welche Klammer referenziert werden soll

{

wenn die geschweifte Klammer auf der letzten Bildschirmzeile steht, wird der Bildschirm so gerollt, daß die korrespondierende Klammer auf der ersten Zeile angezeigt wird

(  
das gleiche mit runden Klammern

)  
dito

[  
diesmal mit eckigen Klammern

]   
auch nichts neues

ESC-CTRL-F

wie {; es werden die dem Kommando folgenden beiden Zeichen als Klammern interpretiert und entsprechend bearbeitet

ESC-CTRL-B

wie }; es werden die dem Kommando folgenden beiden Zeichen als Klammern interpretiert und entsprechend bearbeitet

m

gefolgt von einem Kleinbuchstaben, markiert die aktuelle Position mit diesem Buchstaben

' | CTRL-X CTRL-X

gefolgt von einem Kleinbuchstaben, kehrt zurück zu der mit diesem Buchstaben markierten Stelle

### **/Muster**

sucht vorwärts nach dem *Muster*; eine passende Zeile wird als erste Bildschirmzeile angezeigt; das Muster kann reguläre Ausdrücke enthalten (ed-Syntax); mit einem Zahlenargument N kann auch das N-te Auftreten des Musters gesucht werden; die Suche beginnt in der zweiten Bildschirmzeile, wenn nicht durch die Kommandozeilenoptionen ``-a'` oder ``-j'` ein anderes Verhalten eingestellt ist; für das erste Zeichen vom Muster stehen folgende Sonderzeichen zur Verfügung:

!

sucht nach einer Zeile, die das Muster nicht enthält

\*

durchsucht mehrere Dateien; wenn die Suche bis zum Dateiende erfolglos ist, wird die nächste Datei aus der Kommandozeilenliste durchsucht

@

beginnt die Suche in der ersten Datei aus der Kommandozeilenliste, unabhängig davon, welche Datei aktuell angezeigt wird; auch die Kommandozeilenoptionen ``-a'` und ``-j'` werden ignoriert

### **?Muster**

sucht rückwärts nach einer Zeile mit dem *Muster*; die Suche beginnt vor der ersten Bildschirmzeile; folgende Sonderzeichen können als erstes Zeichen vom Muster eingesetzt werden:

!

sucht eine Zeile, die das Muster nicht enthält

\*



durchsucht mehrere Dateien; wenn das Muster bis zum Dateianfang erfolglos ist, wird die der aktuellen Datei vorhergehende aus der Kommandozeilenliste durchsucht

@

beginnt die Suche auf der letzten Zeile der letzten Datei aus der Kommandozeilenliste, unabhängig von der aktuellen Datei und den Kommandozeilenoptionen ``-a'` und ``-j'`

ESC-/***Muster***

das gleiche wie ``/*Muster'`

ESC-?***Muster***

das gleiche wie ``?*Muster'`

n

wiederholt die letzte Suche; die Sonderzeichen ``*'`` und ``!'`` behalten ihre Bedeutung, ``@'` wird ignoriert

N

wiederholt die letzte Suche in umgekehrter Richtung

ESC-n

wiederholt die letzte Suche und sucht in der nächsten (vorhergehenden) Datei weiter, wenn das Dateende (der Dateianfang) erreicht ist

ESC-N

wiederholt die Suche rückwärts und setzt die Suche in weiteren Dateien fort

:e [***Dateiname*** ...]

zeigt die benannte Datei an; wenn keine Datei angegeben ist, wird die aktuelle Datei neu angezeigt; ein Prozentzeichen ``%'`` im Dateiname wird durch den Namen der aktuellen Datei ersetzt; das Nummernzeichen ``#'` wird durch den Namen der zuvor angezeigten Datei ersetzt; die neue Datei wird in die Kommandozeilenliste der Dateien eingereiht, so daß sie bei den folgenden ``:n'` und ``:p'` Befehlen erreicht werden kann; wenn mehrere Dateinamen angegeben sind, werden alle Dateinamen in die Kommandozeilenliste eingefügt

CTRL-X CTRL-V | E

das gleiche wie ``:e'`

:n

zeigt die nächste Datei aus der Kommandozeilenliste an

:p

zeigt die vorhergehende Datei aus der Kommandozeilenliste an

:X

zeigt die erste (oder N-te) Datei aus der Kommandozeilenliste an

= | CTRL-G | :f

zeigt den Namen der aktuellen Datei und die Position in der Datei

-

,gefolgt von einer der unten aufgeführten Kommandozeilenoptionen, ändert eben diese Option und zeigt die neue Einstellung an; wenn zu einer Option ein Zahlenargument oder eine Zeichenkette angegeben werden müssen, kann dieses Argument nach dem Optionsbuchstaben interaktiv angegeben werden

--+

,gefolgt von einer der unten beschriebenen Kommandozeilenoptionen, setzt diese Option auf ihren voreingestellten Wert zurück; es können nur die Einstellungen zurückgesetzt werden, die keine Zeichenkette als Argument benötigen

-

,gefolgt von einer der unten beschriebenen Kommandozeilenoptionen, setzt diese Option auf das Gegenteil der Standardeinstellung; es können nur die Einstellungen verändert werden, die keine Argumente benötigen; im Anschluß an die Änderung wird die aktuelle Einstellung gezeigt

—

(Unterstrich), gefolgt von einer der unten beschriebenen Kommandozeilenoptionen, zeigt den aktuellen Wert dieser Einstellung; die Einstellung wird nicht verändert

### +**Lesskommando**

führt das angegebene *Lesskommando* jedesmal automatisch aus, wenn eine neue Datei angezeigt wird

V

zeigt die Versionsnummer von `less`

q | :q | :Q | ZZ | ESC ESC

beendet `less`

v

startet einen Editor mit der aktuellen Datei; wenn in der Umgebungsvariablen `EDITOR` nichts anderes bestimmt ist, wird der `vi` als Standardeditor benutzt

### ! **Kommandozeile**

startet eine Shell und führt das angegebene externe Kommando aus; in der *Kommandozeile* kann der Name der aktuellen Datei mit einem Prozentzeichen ``%'`, der Name der zuletzt davor aktuellen Datei mit dem Nummernzeichen ``#'` ersetzt werden; ein doppeltes Ausrufezeichen ``!!'` wiederholt die letzte Kommandozeile; ein einfaches Ausrufezeichen ``!'` startet eine interaktive Shell; in allen Fällen wird die Standardshell `/bin/sh` gestartet, wenn in der Umgebungsvariablen `SHELL` keine andere Shell bestimmt ist

### | **Marke Kommandozeile**

leitet die Zeilen von der ersten Bildschirmzeile bis zur Marke durch das in der *Kommandozeile* angegebene Kommando

## Optionen:

Die folgenden Optionen und Einstellungen können in der Kommandozeile beim Aufruf von `less` gesetzt werden. Es ist außerdem möglich, die entsprechenden Einstellungen mit dem ``-'`-Kommando zur Laufzeit von `less` vorzunehmen. Zusätzlich bietet `less` die Möglichkeit, bestimmte Optionen, die jedesmal gesetzt werden sollen, in der Umgebungsvariablen `LESS` zu speichern. Die so gesetzten Optionen werden jedesmal gelesen, wenn `less` gestartet wird, können aber immer von Kommandozeilenoptionen verdeckt werden.

-?

zeigt eine kurze Übersicht über die Kommandos und Optionen von `less`

-a

die Vorwärtssuchfunktionen fangen in der letzten Bildschirmzeile an, überspringen also den aktuellen Bildschirm

-b **Nr**

veranlaßt `less`, die mit *Nr* angegebene Anzahl von Puffern für die Anzeige zu benutzen; jeder Puffer ist ein Kilobyte groß; wenn `less` aus einer Pipe liest, werden die Puffer automatisch angefordert (siehe die Option ``-B'`)

-B

unterdrückt die automatische Anforderung neuer Puffer; es werden nur die mit der ``-b'` Option bereitgestellten oder die standardmäßig eingestellten zehn Puffer für die Speicherung der aus einer Pipeline gelesenen Daten benutzt; werden mehr Daten gelesen, als Pufferplatz frei ist, wird der älteste Puffer überschrieben

-c

jede neue Bildschirmseite wird von der ersten Zeile an neu aufgebaut; normalerweise wird jede neue Bildschirmseite durch Scrollen des Bildschirms angezeigt

-C

der Bildschirm wird vor jeder neuen Seite gelöscht sonst wie ``-c'`

-d

unterdrückt Warnungen bzw. Fehlermeldungen auf ```dummen"` Terminals, wenn bestimmte Funktionen wie Bildschirmlöschen oder Rückwärtsscrollen nicht zur Verfügung stehen

-e

`less` beendet automatisch, wenn das Dateiende zum zweiten mal erreicht wird; normalerweise kann `less` nur ausdrücklich durch ein entsprechendes Kommando verlassen werden

-E

`less` wird automatisch beim (ersten) Dateiende verlassen

-f

erzwingt die Anzeige, auch von Dateien mit nichtdruckbaren Zeichen

-h **Nr**

es werden höchstens die angegebene Anzahl Zeilen rückwärts gescrollt; sollen mehr als die angegebene Anzahl Zeilen zurückgeblättert werden, findet der Bildschirmaufbau wie beim Vorwärtsblättern von der ersten Zeile an statt

-i

Groß- und Kleinschreibung werden beim Suchen nicht unterschieden; wenn ein Buchstabe im Suchmuster groß geschrieben ist, wird diese Option ignoriert

-j **Nr**

die Zielzeile bei einer Suche oder einer direkten Positionierung wird an der benannten Stelle angezeigt; eine negative Zahl zählt die Zielzeile vom unteren Bildschirmrand aus (anstelle des oberen)

-k **Datei**

liest eine alternative Tastaturbelegung aus der benannten Datei; wenn keine Datei angegeben ist, wird die Datei ``.less'` im Heimatverzeichnis gelesen; eine Datei mit alternativer Tastaturbelegung kann mit dem `lesskey`-Kommando erzeugt werden

-m

zeigt die aktuelle Dateiposition in Bytes oder Prozent als Eingabeaufforderung nach jeder Seite; normalerweise dient der Doppelpunkt ':' als Eingabeaufforderung

-M

zeigt die aktuelle Dateiposition in Zeilen und den Dateinamen als Eingabeaufforderung

-n

die Zeilennummer wird weder in der Eingabeaufforderung, noch beim Aufruf des Editors benutzt

-N

jeder Zeile wird in der Anzeige eine Zeilennummer vorangestellt

-o **Datei**

kopiert die gelesene Eingabe in die benannte *Datei*, wenn aus einer Pipeline gelesen wird; wenn die Datei schon existiert, muß der Benutzer vor dem Überschreiben die Aktion bestätigen

-O **Datei**

wie '-o'; es wird aber vor dem Überschreiben einer existierenden Datei nicht nachgefragt

-p **Muster**

sucht sofort das erste Auftreten vom *Muster* in der ersten angegebenen Datei

-P **Prompt**

ändert die Eingabeaufforderung (*Prompt*); '-Pm' ändert den Prompt für die '-m' Option, und '-PM' ändert entsprechend den Prompt für die '-M' Option; die Syntax der Prompt Zeichenkette mit 'man less' in den englischen Manualpages nachzulesen

-q

der Alarmton des Terminals wird nicht bei kleineren Fehlern wie z. B. das Vorwärtsblättern am Dateiende ausgelöst; wenn das Terminal eine optische Warnung unterstützt, wird diese anstelle des akustischen Alarms benutzt

-Q

der Alarmton des Terminals wird unter keinen Umständen ausgelöst

-r

die Sonderzeichen (Control Sequenzen) werden nicht als Caret-Sequenz, sondern 'roh' angezeigt

-s

mehrere Leerzeilen werden zu einer einzigen Leerzeile zusammengefaßt; das ist die Standardeinstellung für die Anzeige von *groff*-Ausgabe

-S

überlange Zeilen werden einfach abgeschnitten; normalerweise wird der Rest überlanger Zeilen in der nächsten Zeile angezeigt

-t **Tag**

zeigt die Datei, in der vom *ctags*-Programm die Marke *Tag* gefunden wurde; für diese Option muß eine 'tags' Datei vom *ctags* Programm im aktuellen Verzeichnis angelegt werden

-T **Tagdatei**

die benannte Tagdatei wird anstelle der Datei 'tags' im aktuellen Verzeichnis für die '-t' Option benutzt

-u

Rückschritt und Wagenrücklauf werden roh (unverändert) an das Terminal geschickt

-U

Rückschritt und Wagenrücklauf werden als Control-Sequenzen der Option ``-r'` entsprechend an das Terminal geschickt; normalerweise werden Rückschritte, die von einem Unterstrich gefolgt werden, durch einen unterstrichenen Buchstaben, Rückschritte, die von dem gleichen Buchstaben eingeleitet wie gefolgt werden, durch einen fettgedruckten Buchstaben dargestellt; alle anderen Rückschritte und die von einem Zeilenvorschub gefolgt werden Wagenrücklaufzeichen werden ignoriert

-w

die Bildschirmzeilen nach dem Dateiende werden durch Leerzeilen anstelle der voreingestellten Tilde ``~'` dargestellt

-x *Nr*

die Tabulatorweite wird auf die gegebene Anzahl Stellen gesetzt

-y *Nr*

setzt eine Grenze von Zeilen, bis zu der das Vorwärtsscrollen möglich ist; wenn mehr als die gesetzte Anzahl Zeilen vorwärts geblättert werden soll, wird der Bildschirm der ``-c'` oder ``-C'` Option entsprechend neu aufgebaut

-z *Nr*

ändert die Schrittweite für das Blättern auf die angegebene Zeilenzahl

+

eine Kommandozeilenoption, die mit einem Pluszeichen beginnt, wird als Initialisierungskommando für `less` interpretiert und als erstes Kommando von `less` ausgeführt; wenn anstelle eines einzelnen Pluszeichens zwei Pluszeichen benutzt werden, wird das darauffolgende Kommando bei allen Dateien der Kommandozeilenliste als erstes ausgeführt

## Umgebungsvariable

`less` unterstützt die folgenden Umgebungsvariablen:

**COLUMNS**

die Zeilenlänge für den Ausgabebildschirm

**EDITOR**

der Editor für das ``v'`-Kommando

**HOME**

das Heimatverzeichnis des Anwenders; hier wird die ``.less'` Datei gesucht

**LESS**

eine Zeichenkette mit Kommandozeilenoptionen

**LESSBINFMT**

das Format zur Anzeige binärer Zeichen, die keine Controlzeichen sind

**LESSCHARDEF**

legt einen Zeichensatz fest (ascii oder latin1)

## LESSCHARSET

beschreibt einen anderen Zeichensatz direkt

## LESSEDIT

die Kommandozeile für den Editor

## LESSHELP

der absolute Name der Hilfsdatei

## LINES

die Anzahl der Bildschirmzeilen

## SHELL

die Shell für die externen Kommandos

## TERM

die Terminalbezeichnung, wie sie in /etc/termcap zu finden ist

## Siehe auch:

[more](#) und [lesskey\(1\)](#)

## Autor:

Mark Nudelman

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [ln](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [kill](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

**Next:** [login](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [less](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# In

## Funktion:

**ln** (link) erzeugt einen Verzeichniseintrag einer existierenden Datei unter anderem Namen

## Syntax:

```
ln [Optionen] Quelle [Ziel]
```

```
ln [Optionen] Quelle ... Zielverzeichnis
```

## Beschreibung:

Jede Datei wird bei ihrer Erzeugung mit ihrem Namen in ein Verzeichnis eingetragen. Dieser Eintrag enthält außerdem einen Verweis auf eine Inode, in der die Zugriffsrechte auf die Datei, der Dateityp und gegebenenfalls die Nummern der belegten Datenblöcke eingetragen sind.

Mit dem **ln**-Kommando wird ein neuer Eintrag in einem Verzeichnis angelegt, der auf die Inode einer existierenden Datei zeigt. Diese Art Link wird als 'Hardlink' bezeichnet. Weil die Zugriffsrechte auf die Datei in der Inode bestimmt werden, sind die Zugriffsrechte auf alle Links einer Datei gleich.

Sie müssen dem **ln**-Kommando nach den Optionen zuerst den Namen der existierenden Quelldatei(en) angeben und danach den Namen des gewünschten zusätzlichen Eintrags.

Ist das letzte Argument des Aufrufs ein existierendes Verzeichnis, so werden alle als *Quelle* aufgelisteten Dateien unter dem gleichen Namen im *Zielverzeichnis* in diesem Verzeichnis eingetragen.

Wird nur eine einzige Quelldatei ohne eine Zieldatei oder ein Zielverzeichnis benannt, so wird ein Link unter diesem Namen im aktuellen Verzeichnis angelegt.

Hardlinks können nur auf dem Datenträger angelegt werden, auf dem sich die Datei (und damit die Inode) selbst befindet. Um Links über die Dateisystemgrenzen hinweg anlegen zu können, bietet Linux die Möglichkeit 'symbolischer Links'. In diesen Links ist der absolute Pfad gespeichert, auf dem die gelinkte Datei gefunden werden kann. Ein Zugriff auf diese Datei wird dann vom

Betriebssystem automatisch auf die gelinkte Datei umgelenkt. Im ext2fs können symbolische Links mit Pfadnamen bis zu einer Länge von 60 Zeichen in der Inode selbst gespeichert werden (fast symbolic link). In allen anderen Fällen wird für den Link ein Datenblock auf der Festplatte belegt. Normalerweise löscht `ln` keine existierenden Dateien. Es werden standardmäßig ``hardlinks" angelegt. Links auf Verzeichnisse oder auf Dateien in anderen Dateisystemen können nur mit symbolischen Links realisiert werden.

Gelegentlich verändert sich das Verhalten eines Programms, wenn es durch einen Link unter einem anderen Namen aufgerufen wird. (Das funktioniert natürlich nur, wenn diese Änderung im Programm vorgesehen ist.)

## Optionen:

- b  
sichert Dateien, anstatt sie zu löschen (mit Option -f)
- d  
unter Linux ohne Funktion
- f  
löscht bestehende Dateien
- i  
fragt vor dem Löschen nach Bestätigung
- s  
macht symbolische Links anstelle von harten
- v  
gibt die Dateinamen auf den Bildschirm
- S **Endung**  
setzt die Endung für die Sicherung von Dateien auf *Endung*. Standardwert ist `~'. Die Endung kann auch mit der Umgebungsvariablen `SIMPLE_BACKUP_SUFFIX` bestimmt werden. Die Option -S hat Vorrang vor der Umgebungsvariablen.
- V {numbered, existing, simple}  
bestimmt die Art der Sicherungskopien. Die Art der Sicherung kann auch mit der Umgebungsvariablen `VERSION_CONTROL` bestimmt werden. Die Option -V hat auch hier die höhere Priorität. Die Optionen bedeuten hierbei:
  - numbered  
macht immer numerierte Backups
  - existing  
macht numerierte Backups nur für bereits numerierte Dateien
  - simple  
macht immer nur einfache Backups

## Autor:

Mike Parker und David MacKenzie

---





## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# login

## Funktion:

`login` prüft die Identität des Benutzers und startet eine Shell

## Syntax:

`login` [*Name* ]

## Beschreibung:

Der Zugang zum Linux-Systemen ist normalerweise nur eingetragenen Benutzern möglich. Diesen Anwendern ist dann die Ausführung von Programmen und das Lesen bzw. Schreiben von Dateien in dem Umfang möglich, wie es die [Zugriffsrechte](#) im Dateisystem erlauben. Damit kann ein Unix-Rechner weitgehend vor Fehlern oder gar Mißbrauch geschützt werden. Um die Identität eines Benutzers festzustellen, wird zu Beginn jeder „Sitzung“ ein `login` durchgeführt, bei dem zu dem Benutzernamen noch ein Paßwort abgefragt wird. Erst wenn das richtige Paßwort eingegeben ist, läßt sich das System benutzen. Um während einer Sitzung die Rechte eines anderen Users (z. B. **root** für Verwaltungszwecke) zu erhalten, kann außer dem Kommando [su](#) auch ein `login` wie jeder andere Befehl in der Shell aufgerufen werden. Man erhält nach dem `login` eine neue Shell mit den Rechten des Users. Nach Verlassen dieser Shell mit `exit` ist man wieder in der eigenen Shell mit den eigenen Rechten.

## Optionen:

### *Name*

es wird ein `login` für den Anwender *Name* durchgeführt

## Autor:

Michael Glad

---



**Next:** [ls](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [login](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
- 

# logname

## Funktion:

**logname** zeigt den Benutzernamen

## Syntax:

**logname**

## Beschreibung:

Das Kommando **logname** zeigt den Benutzernamen, wie er von `getty` in der Datei `/var/run/utmp` gespeichert ist. Wenn für das Terminal, von dem aus das Kommando gestartet wird, kein Eintrag gefunden wird, gibt `logname` eine Fehlermeldung aus und beendet mit dem Status 1; sonst wird mit dem Status 0 beendet.

Weil nach dem `getty` bis zum Ausloggen des Benutzers kein Programm den Eintrag für ein bestimmtes Terminal in der `/var/run/utmp` Datenbasis verändert, wird mit dem `logname`-Kommando immer der Name angezeigt, unter dem sich ein Systembenutzer ursprünglich im System angemeldet hat, unabhängig davon ob er aktuell zum Beispiel nach einem `su`-Kommando unter einer anderen Benutzerkennung arbeitet.

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# ls

## Funktion:

**ls** (list) zeigt den Inhalt eines Verzeichnisses

## Syntax:

```
ls [-abcdgiklmnpqrstuxABCFLNQRSUXl] [-w Spalten] [-T Spalten] [-I
Muster] [-all] [-escape] [-directory] [-inode] [-kilobytes]
[-numeric-uid-gid] [-hide-control-chars] [-reverse] [-width=Spalten]
[-tabsize=Spalten] [-size] [-almost-all] [-ignore-backups]
[-classify] [-file-type] [-ignore=Muster] [-dereference] [-literal]
[-quote-name] [-recursive] [-sort={none, time, size, extension}]
[-format={long, verbose, commas, across, vertical, single-column}]
[-time={atime, access, use, ctime, status}] [Pfad ...]
```

## Beschreibung:

**ls** zeigt den Inhalt der Verzeichnisse des Dateisystems an.

Das Standardausgabeformat von **ls** hängt vom Typ der Ausgabedatei ab. Auf einem Terminal ist die mehrspaltige Ausgabe das Standardformat. In allen anderen Fällen wird die Ausgabe einspaltig ausgeführt.

Das Verhalten des **ls**-Kommandos läßt sich nicht mehr durch Umbenennen in **ll** **dir** **vdir** etc. verändern. Stattdessen sind die Kommandos **dir** und **vdir** als separate Binärdateien mit entsprechenden Standardformaten verfügbar.

## Optionen:

-a

zeigt alle Dateien im Verzeichnis, auch die deren Name mit '.' beginnt

-b

zeigt nichtdruckbare Zeichen in Dateinamen als „Backslash Sequenz“ mit alphabetischen oder oktalen Werten, wie sie in C üblich sind

-c

sortiert die Dateien nach der Zeit der letzten Statusveränderung

-d

zeigt Unterverzeichnisse wie normale Dateien anstelle ihres Inhaltes

-i

zeigt die Nummer der Inode zu jeder Datei

-k

die Dateigröße wird in Kilobytes angegeben, auch wenn POSIXLY\_CORRECT gesetzt ist

-l

außer dem Namen werden der Typ, die Rechte, die Anzahl der Hardlinks, der Besitzer, die Gruppe, die Größe und die Zeitmarke angezeigt

-m

gibt die Dateinamen in einer Reihe, getrennt durch Kommas aus

-n

gibt die Benutzer und Gruppen mit ihren ID's anstelle der Namen aus

-q

gibt Fragezeichen anstelle von nicht druckbaren Zeichen in Dateinamen

-r

zeigt das Verzeichnis in umgekehrter Reihenfolge

-s

zeigt die Größe der Dateien in Kilobytes; wenn POSIXLY\_CORRECT gesetzt ist, wird die Größe in Blöcken zu 512 Bytes angezeigt

-t

sortiert nach Zeit anstelle des Namens

-u

sortiert nach letzter Zugriffszeit anstelle der Änderungszeit (zusammen mit Option -t)

-x

sortiert in horizontaler Richtung

-A

zeigt alle Dateien außer `.` und `..`

-B

ignoriert Backups (mit Endung `~`)

-C

listet in vertikal sortierten Spalten

-F

hängt verschiedene Symbole an die Dateinamen:

\*

steht hinter ausführbaren Dateien

/

steht hinter Verzeichnissen

@

markiert symbolische Links

|

markiert FiFo's

=

markiert sockets

alles andere sind reguläre Dateien

-L

zeigt den Inhalt der symbolisch gelinkten Verzeichnisse anstelle des Linkfiles

-N

gibt Dateinamen ohne Quotes aus

-Q

gibt Dateinamen in Quotes aus

-R

zeigt rekursiv den Inhalt aller Unterverzeichnisse

-S

sortiert nach Größe

-U

unsortiert

-X

sortiert nach Endung

-l

einspaltig

-w **Spalten**

Bildschirmbreite in *Spalten*

-T **Spalten**

Tabulatorbreite in *Spalten*

-I **Muster**

ignoriert Dateien mit *Muster* im Namen

## Autor:

Richard Stallman und David MacKenzie

---

[Next](#) [Up](#) [Previous](#) [Contents](#) [Index](#)

**Next:** [man](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [logname](#)





## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- 

# man

## Funktion:

**man** zeigt die Manualpages zu den Linux-Kommandos

## Syntax:

**man** [-adfhktw] [-m *System*] [-p *Zeichenkette*] [-M *Pfad*] [-P *Pager*] [-S *List*] [*Section*] *Name* ...

## Beschreibung:

**man** gibt die Manualpages zum Thema *Name* aus. Diese englischen Handbuchseiten sind Teil des Linux-Systems. Wenn die Seiten im rohen nroff-Format vorliegen, werden sie automatisch formatiert. Die Anzeige erfolgt durch einen „Pager“, der immer nur eine Bildschirmseite zur Zeit anzeigt. Solche „Pager“ stehen als eigenständige Programme zur Verfügung. Als Standardpager wird `less` benutzt, in der Umgebungsvariablen `PAGER` kann aber ein anderes Programm bestimmt werden.

Die Manualpages werden in verschiedene Kapitel unterteilt:

1. die Benutzerkommandos
2. die Systemaufrufe
3. die C-Bibliotheksfunktionen
4. die Beschreibungen der Gerätedateien
5. die Dateiformate

6. Spiele
7. die Makropakete für die Textformatierer
8. die Kommandos für die Systemverwalterin
9. für die Kernelroutinen

Außerdem werden noch die Sektionen ``l'` (lokale) ``p'` (private) ``n'` (new) und ``o'` (old) und `tbl` unterstützt.

Die Reihenfolge, in der die Sektionen nach einer bestimmten Manualpage durchsucht werden, ist in der Konfigurationsdatei `/etc/man.config` festgelegt. In der `MANSEC` Umgebungsvariablen kann jeder User für sich eine andere Reihenfolge bestimmen.

Die Verzeichnisse, in denen man nach Manualpages sucht, werden von der Systemverwalterin ebenfalls in der Datei `/etc/man.config` bestimmt. In der `MANPATH` Umgebungsvariablen können andere Verzeichnisse angegeben werden.

## Optionen:

- `-M Pfad`  
verdeckt die `MANPATH` Umgebungsvariable; auf dem *Pfad* wird nach den Manualpages gesucht
- `-P Pager`  
bestimmt das externe Programm *Pager* als Anzeigefilter
- `-S Liste`  
*Liste* ist eine durch Komma getrennte Liste von Kapiteln (Sektionen) des Manuals; die *Liste* verdeckt die `MANSECT` Umgebungsvariable
- `-a`  
zeigt alle Manualpages, die auf dem *Pfad* gefunden werden
- `-d`  
gibt Fehlerinformationen zum Entwanzen
- `-f`  
wie `what is`
- `-h`  
gibt eine Hilfszeile zum `man`-Kommando aus
- `-k`  
wie `apropos`
- `-p Namen`  
gibt die Präprozessoren für `groff` an
- `-t`  
formatiert die Manualpages mit dem Kommando ``groff -Tascii -mandoc'` und leitet das Ergebnis auf die Standardausgabe

–w

gibt nur den Pfad der Manualpages aus, nicht deren Inhalt

## Siehe auch:

das `info`-Kommando für das T<sub>E</sub>Xinfo System und das Shellkommando [help](#)

### Autor:

John W. Eaton

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [mcopy](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [ls](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- 

# mcopy

## Funktion:

**mcopy** kopiert Dateien von/nach DOS-Filesystemen nach/von Linux-Dateisystemen

## Syntax:

**mcopy** [ -tnwm] *Quelldatei* *Zieldatei*

**mcopy** [ -tnwm] *Quelldatei* [*Quelldatei* ...] *Zielverzeichnis*

## Beschreibung:

**mcopy** kopiert Dateien zwischen MS-DOS und Linux-Dateisystemen. Ein MS-DOS Dateiname muß mit einer Laufwerksbezeichnung angegeben werden. Wird für eine MS-DOS Datei kein absoluter Pfad benannt, gilt das aktuelle DOS-Verzeichnis. Dieses Verzeichnis wird in der Datei `~/.mcwd' im Heimatverzeichnis des Anwenders bestimmt und durch das `mcd`-Kommando verändert.

## Optionen:

-t

konvertiert CR/LF in LF (für Texttransfer)

-n

unterdrückt Warnungen vor dem Überschreiben existierender Dateien

-v

gibt die Dateinamen vor dem Kopieren aus

-m

kopiert die Datei inclusive des Zeitstempels

## Siehe auch:

`mcd(1)`, [mread](#), `mread(1)`, `mwrite(1)`, `/etc/mtools`, `~/.mcwd` und [mtools](#)

---

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

**Next:** [mdir](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [mcopy](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- 

# mdel

## Funktion:

**mdel** löscht eine Datei in einem MS-DOS Dateisystem

## Syntax:

```
mdel [-v] msdosdatei ...
```

## Beschreibung:

**mdel** löscht eine oder mehrere Dateien in einem MS-DOS Dateisystem. **mdel** erwartet eine Bestätigung vor dem Löschen von „nur Lesen“ Dateien.

## Optionen:

**-v**

zeigt den Namen jeder Datei vor dem Löschen noch einmal an

## Siehe auch:

[mtools](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- 

# mdir

## Funktion:

**mdir** zeigt den Inhalt eines DOS Verzeichnisses

## Syntax:

**mdir** [-w] [*Verzeichnis*]

**mdir** [-w] *Datei* ...

## Beschreibung:

**mdir** zeigt den Inhalt eines MS-DOS Verzeichnisses oder Information über MS-DOS Dateien. Wird kein *Verzeichnis* angegeben, so wird der Inhalt des aktuellen DOS Verzeichnisses angezeigt, das in der Datei `~/mcwd` im Heimatverzeichnis des Anwenders festgelegt ist und mit dem `mcd`-Kommando gewechselt wird.

## Optionen:

-w

gibt die Dateinamen in Spalten nebeneinander aus

## Siehe auch:

[mtools](#)

**Next:** [mkdir](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [mdir](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- 

# mformat

## Funktion:

**mformat** richtet ein MS-DOS Dateisystem ein

## Syntax:

**mformat** [ *-t Spuren*] [ *-h Köpfe*] [ *-s Sektoren*] [ *-l Label*] *Laufwerk*:

## Beschreibung:

Das **mformat**-Kommando gehört zu den `mttools` zur Bearbeitung von MS-DOS Disketten und richtet auf einer roh (low-level) formatierten Diskette ein MS-DOS Dateisystem ein. So eine Diskette kann entweder mit den übrigen `mttools` bearbeitet werden oder mit dem `mount`-Kommando in das Linux-Dateisystem eingebunden werden.

Die Standardparameter für Spuren/Köpfe/Sektoren werden aus der Datei `/etc/mttools` gelesen, sie können aber auch auf der Kommandozeile eingegeben werden.

## Optionen:

**-t *Anzahl***

die Anzahl der Spuren

**-h *Anzahl***

die Anzahl der Köpfe (Seiten)

**-s *Anzahl***

die Anzahl der Sektoren pro Spur

**-l *Name***

das Label der Diskette



## Siehe auch:

[fdformat](#) und [mtools](#)

---

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

**Next:** [mkfifo](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [mformat](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- 

# mkdir

## Funktion:

**mkdir** erzeugt ein leeres Verzeichnis

## Syntax:

**mkdir** [-p] [-m *Modus*] [-path] [-mode=*Modus*] *Verzeichnis* ...

## Optionen:

-m *Modus*

setzt die Rechte des Verzeichnisses auf *Modus*; der *Modus* wird wie beim Kommando [chmod](#) angegeben; der Standard, und damit der Ausgangswert für relative Modes, ist 0777 minus der Bits von umask (siehe bei [umask](#))

-p

wenn ein Unterverzeichnis in einem nicht existierenden Verzeichnis angelegt werden soll, werden alle fehlenden Verzeichnisse angelegt

## Siehe auch:

[ln](#) und [rmdir](#)

## Autor:

David MacKenzie

Next: [mmd](#) Up: [Von GNU's, Muscheln und](#) Previous: [mkdir](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- 

# mkfifo

## Funktion:

**mkfifo** erzeugt eine FIFO-Datei

## Syntax:

**mkfifo** [ *-m Modus* ] [ *-mode=Modus* ]

## Beschreibung:

**mkfifo** erzeugt eine FIFO-Datei (Named Pipe). Daten die in diese Datei geschrieben werden, können nur sequentiell in der gleichen Reihenfolge wieder gelesen werden. FIFO steht für „First In First Out. Die Zugriffsrechte auf die Datei werden aus der Bitdifferenz zwischen 0666 und dem Wert von [umask](#) errechnet, wenn nicht ausdrücklich ein anderer Modus angegeben ist.

## Optionen:

*-m Modus*

setzt bzw. ändert die Zugriffsrechte der Datei; *Modus* ist dabei wie bei `chmod` beschrieben

## Siehe auch:

[mknod](#)

## Autor:

David MacKenzie



## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- 

# mmd

## Funktion:

**mmd** erzeugt ein neues MS-DOS Verzeichnis

## Syntax:

**mmd** [-v] *Verzeichnis* ...

## Beschreibung:

**mmd** erzeugt ein oder mehrere neue MS-DOS Verzeichnisse im aktuellen DOS-Verzeichnis oder in dem absoluten Pfad. Verzeichnisnamen, die nicht der MS-DOS Konvention entsprechen werden automatisch auf die zulässigen Zeichen gekürzt. Existiert bereits eine Datei oder ein Verzeichnis mit dem gleichen Namen, wird das Kommando nicht ausgeführt.

## Optionen:

-v  
zeigt die Namen der erzeugten Verzeichnisse an

## Siehe auch:

[mtools](#)

**Next:** [mrd](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [mmd](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# more

## Funktion:

**more** zeigt Dateien seitenweise

## Syntax:

**more** [ -cdflsu ] [ -n ] [ +linenumber ] [ +/pattern ] [*Name* ...]

## Beschreibung:

**more** gibt Textdateien seitenweise auf dem Bildschirm aus. Nach jeder Bildschirmseite wird die Ausgabe angehalten und auf eine Eingabe des Benutzers gewartet.

In der Umgebungsvariablen MORE können Kommandozeilenoptionen für **more** gespeichert werden, die bei jedem Aufruf automatisch ausgeführt werden.

Am Dateiende wird **more** automatisch beendet.

Wenn **more** eine Datei liest, wird auf der letzten Bildschirmzeile im 'Prompt' die prozentuale Position der aktuellen Bildschirmseite in der Datei angezeigt.

Wenn dieser Prompt angezeigt wird, erwartet **more** eine Eingabe des Benutzers. Die Eingaben bestehen in der Regel aus einem einzigen Tastendruck. Einige Kommandos können mit einer Zahlenangabe *N* kombiniert werden.

*N*LEERZEICHEN

gibt die nächsten *N* Zeilen aus oder einen kompletten Bildschirm, wenn keine Zahl angegeben ist

CTRL-D

gibt die nächsten 11 Zeilen (einen halben Bildschirm) aus

d

das gleiche wie ^D

**Nz**

das gleiche wie Leerzeichen; die Zahl *N* wird die neue Anzahl Zeilen pro Bildschirm

**Ns**

überspringt die nächsten *N* Zeilen und zeigt die darauffolgenden Zeilen an

**Nf**

überspringt die nächsten *N* Bildschirme und zeigt die darauffolgenden Zeilen an

**Nb**

springt *N* Bildschirmseiten rückwärts

**N<sub>CTRL</sub>-B**

das gleiche wie **b**

**q** oder **Q**

beendet **more**

**=**

zeigt die aktuelle Zeilennummer an

**v**

startet den Editor **vi** mit der aktuellen Datei in der aktuellen Zeile

**h**

das Hilfefkommando; gibt eine Übersicht über die **more**-Kommandos

**N/Ausdruck**

sucht das *N*te Auftreten des Ausdrucks vom aktuellen Bildschirm an vorwärts

**N n**

sucht das *N*te aufreten des zuletzt gesuchten Ausdrucks vorwärts

**,**

geht zurück an die Position, von der das letzte Suchkommando gestartet wurde

**! Kommandozeile**

startet eine Shell und führt die angegebene Kommandozeile aus

**N:n**

springt zur nächsten Datei aus der Kommandozeilenliste bzw. *N* Dateien weiter

**N:p**

springt an den Anfang der aktuellen Datei oder in die vorhergehende Datei aus der Kommandozeilenliste bzw. *N* Dateien zurück

**:f**

zeigt den Namen der aktuellen Datei und die Position des aktuellen Bildschirms

**:q** oder **:Q**

beendet **more**

**.**

(Punkt) wiederholt das letzte Kommando

## Optionen:

-N

N ist eine ganze Zahl und setzt die Zeilenanzahl für den Bildschirm

-c

veranlaßt `more`, den Bildschirm beim Weiterblättern von oben nach unten neu aufzubauen, indem jede Zeile unmittelbar vor dem Überschreiben gelöscht wird; diese Option funktioniert nur auf Terminals, die das Löschen einzelner Zeilen unterstützen

-d

gibt einen längeren Prompt mit zusätzlicher Hilfe für den Anwender aus

-f

es werden die Textzeilen anstelle der Bildschirmzeilen angezeigt; dadurch werden Zeilen mit Controlsequenzen, wie sie z. B. von `groff` erzeugt werden, korrekt in einer Zeile angezeigt

-l

ignoriert `^L` (Seitenvorschub); standardmäßig wird die Ausgabe bei jedem Seitenvorschub angehalten; ein Seitenvorschub am Anfang des Textes bewirkt ein Löschen des Bildschirms

-s

zeigt mehrere Leerzeilen in Folge als eine einzige an

-u

unterdrückt die Behandlung von unterstrichenem Text

+**Zeilennummer**

beginnt die Ausgabe bei *Zeilennummer*

+**Muster**

beginnt die Ausgabe zwei Zeilen vor dem ersten Auftreten von *Muster*

## Autor:

Eric Shienbrood, Geoff Peck, John Foderaro

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [mrd](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [mmd](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)



**Next:** [mread](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [more](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Siehe auch:](#)
- 

# mrd

## Funktion:

**mrd** löscht ein MS-DOS Verzeichnis

## Syntax:

**mrd** *Verzeichnis* ...

## Beschreibung:

**mrd** löscht ein oder mehrere MS-DOS Verzeichnisse. Das Kommando wird mit Fehlermeldung abgebrochen, wenn ein Verzeichnis nicht leer ist.

## Siehe auch:

[mtools](#)

Next: [mt](#) Up: [Von GNU's, Muscheln und](#) Previous: [mrd](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- 

# mread

## Funktion:

**mread** liest eine Datei aus einem MS-DOS Dateisystem und schreibt (kopiert) sie in ein Linux-Dateisystem

## Syntax:

```
mread [-tnm] Msdosdatei Unixdatei
```

```
mread [-tnm] Msdosdatei ...Unixverzeichnis
```

## Beschreibung:

**mread** liest eine *MS-DOS Datei* und kopiert sie in eine *Unixdatei*. Wird anstelle einer *Unixdatei* ein *Unixverzeichnis* angegeben, so werden alle MS-DOS Dateien in das *Unixverzeichnis* kopiert.

## Optionen:

-t

übersetzt CR/LF in LF

-n

unterdrückt Warnung vor dem Überschreiben einer Datei

-m

kopiert den Zeitstempel der *msdosdatei*

## Siehe auch:

[mtools](#)

---

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

**Next:** [mtools](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [mread](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Beispiel:](#)
  - [Siehe auch:](#)
- 

# mt

## Funktion:

**mt** steuert die Operation von Magnetbandgeräten

## Syntax:

**mt** [-V] [-f *Gerätedatei*] [-file=*Gerätedatei*] [-version] *Operation* [*Anzahl*]

## Beschreibung:

**mt** ermöglicht die direkte Bedienung von Magnetbandgeräten.

Die Programme, die unter Linux häufig zum Lesen und Beschreiben von Magnetbändern benutzt werden, wie zum Beispiel `tar`, `cpio` oder `afio`, sind nicht auf den Betrieb mit Magnetbändern spezialisiert. Obwohl sie auch in Zusammenarbeit mit Magnetbandgeräten ihre spezielle Aufgabe erfüllen, wird zur Ansteuerung spezieller Magnetbandfunktionen das `mt`-Kommando benötigt.

Mit `mt` können Magnetbänder beispielsweise zurückgespult, positioniert oder gelöscht werden. Die wichtigsten Bandoperationen, die mit `mt` ausgeführt werden können sind:

`eom`

spult das Band zum Ende der letzten Datei auf dem Magnetband. Dabei ist es unerheblich, wie viele Dateien bereits auf dem Band gespeichert sind.

`fsf` *Anzahl*

spult das Band über die angegebene Anzahl Dateiendemarken vorwärts. Nach dieser Operation steht das Band

`rewind`

spult das Band zurück an den Anfang.

status

ermittelt die Statusinformation vom Magnetbandlaufwerk und gibt die Daten auf den Standardausgabekanal.

erase

löscht und initialisiert das Magnetband.

retension

spult das Band einmal ans Ende und wieder zurück an den Anfang, um es neu zu spannen.

offline

schaltet den Gerätetreiber ab und veranlaßt das Magnetbandlaufwerk gegebenenfalls den Mechanismus zum Bänderauswurf zu betätigen.

asf **Anzahl**

spult das Band zurück und anschließend über die angegebene Anzahl Dateiendemarken vorwärts. Dies ist keine Operation, die direkt vom Bandgerät angeboten wird. Sie wird vom GNU-mt durch Kombination von `rewind` und `fsf` realisiert.

Die meisten der oben aufgeführten Operationen können nur dann sinnvoll eingesetzt werden, wenn das Band im "No Rewind On Close" Modus benutzt wird.

Eine Reihe weiterer Operationen zum Positionieren des Magnetbandes sind zwar bei den meisten Treibern implementiert, das genaue Verhalten ist aber sowohl von den Gerätetreibern als auch von der Hardware abhängig. Sie sollten nach Möglichkeit darauf verzichten, ein Magnetband rückwärts zu positionieren.

bsf **Anzahl**

spult das Band über die angegebene Anzahl Dateiendemarken rückwärts. Bei SCSI-Streamern ist nach dieser Operation keine zuverlässige Aussage über die aktuelle Blocknummer mehr möglich.

fsr **Anzahl**

spult das Band die angegebene Anzahl Datenblöcke vorwärts.

bsr **Anzahl**

spult das Band die angegebene Anzahl Datenblöcke rückwärts.

fsfm **Anzahl**

spult das Magnetband vorwärts vor die angegebene Dateiendemarke.

bsfm **Anzahl**

spult das Magnetband rückwärts vor die angegebene Dateiendemarke.

eof **Anzahl**

erzeugt die angegebene Anzahl von Dateiendemarken an der aktuellen Bandposition.

Speziell für die Steuerung von SCSI-Streamern gibt es eine Version von `mt`, die nicht auf den GNU-Sourcen sondern auf Net-BSD basiert. Diese Version ermöglicht noch weitere, SCSI-spezifische Operationen:

seek **Blocknummer**

positioniert das Band auf die angegebene Blocknummer. Diese Funktion wird nur von einigen Bandgeräten unterstützt.

tell

gibt die Blocknummer an der aktuellen Bandposition aus. Diese Funktion wird nur von einigen Bandgeräten unterstützt. Durch einige Bandoperationen kann der interne Blockzähler ungültig werden.

#### setblk *Blockgröße*

stellt die Blockgröße der physikalischen Datenblöcke ein. Der zulässige Bereich hängt sowohl vom Laufwerk als auch vom Medium ab. Die maximale Blockgröße, die vom Gerätetreiber verarbeitet werden kann, beträgt 32 Kilobyte. Bei einigen Kombinationen von Gerät und Medium ist keine Veränderung der Blockgröße möglich. Um das Bandgerät auf variable Blockgröße einzustellen, müssen Sie als Blockgröße 0 wählen.

#### setdensity *Dichte*

stellt einen neuen Wert für die Aufzeichnungsdichte ein. Die meisten Bandgeräte ignorieren den Versuch, die Aufzeichnungsdichte zu verändern.

#### drvbuffer {0|1}

schaltet den Datenpuffer des Bandgerätes ein oder aus.

## Optionen:

-V

zeigt die Versionsnummer des mt-Kommandos an

-f *Gerätedatei*

gibt das Gerät an, mit dem mt arbeiten soll

## Beispiel:

Sie finden Beispiele zur Benutzung von mt im Kapitel über [Datensicherung](#).

## Siehe auch:

[tar](#), [cpio](#) und das Kapitel über [Datensicherung](#)

## Autor:

David MacKenzie (GNU) Kai Makisara (Linuxversion von Net-BSD)

---

Next	Up	Previous	Contents	Index
------	----	----------	----------	-------

**Next:** [mtools](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [mread](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

**Next:** [mv](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [mt](#)

### Subsections

- [Funktion:](#)
  - [Beschreibung:](#)
- 

# mtools

## Funktion:

**mtools** ist eine Sammlung von Werkzeugen zur Bearbeitung von MS-DOS Dateisystemen auf Diskette oder Festplatte.

## Beschreibung:

Die **mtools** von Emmet P. Gray ermöglichen alle Datei- und Verzeichnisoperationen auf MS-DOS Disketten. Von besonderer Bedeutung ist dabei, daß die Disketten nicht in das Dateisystem eingebunden werden, also kein **mount**-Kommando ausgeführt werden muß.

Unter der Sammelbezeichnung **mtools** verbergen sich dreizehn verschiedene Programme (Tools), von denen jedes ein MS-DOS Vorbild hat.

mattrib	[{+,-}ahrs] <i>Datei</i>	ändert die MS-DOS Dateiattribute
mcd	<i>Verzeichnis</i>	wechselt das Arbeitsverzeichnis auf der DOS Diskette
mcopy	[-tnrv] <i>Quelle Ziel</i>	kopiert DOS Dateien von/nach Linux. Besonders interessant ist die automatische Konvertierung von CR/LF in LF mit der Option -t
mdel	[-v] <i>Datei</i>	löscht eine DOS Datei
mdir	[-w] <i>Datei</i>	zeigt den Inhalt eines DOS Verzeichnisses
mformat		legt ein DOS Dateisystem auf einer Low-Level formatierten Diskette an.
mlabel	[-v] <i>Laufwerk</i>	erzeugt ein Volume-Label für eine DOS Diskette.
mund	[-v] <i>Verzeichnis</i>	legt ein Verzeichnis auf einer DOS Diskette an.
mrd	<i>Verzeichnis</i>	löscht ein Verzeichnis auf einer DOS Diskette.
mread	[-tnm] <i>Quelle Ziel</i>	liest (kopiert) eine Datei "roh" von DOS nach Linux.
mren	[-v] <i>Alt Neu</i>	benennt eine existierende DOS Datei um.
mtype	[-ts] <i>Datei</i>	gibt den Inhalt einer DOS Datei aus.
mwwrite	[-tnvm] <i>Quelle Ziel</i>	schreibt (kopiert) eine Datei "roh" von Linux auf eine DOS Diskette.

Genaue Beschreibungen der `mttools` können Sie in den Manualpages zu den Programmen nachlesen.

Dateinamen und Pfade auf MS-DOS Dateisystemen müssen mit einer Laufwerksbezeichnung beginnen. Diese „Laufwerke“ müssen mit den entsprechenden Gerätedateien in der Datei `/etc/mttools'` verknüpft werden.

Im Unterschied zu den Originalkommandos kann bei den `mttools` als Trenner von DOS-Verzeichnisnamen sowohl ``\`` als auch ``/'` verwendet werden. Die `mttools` akzeptieren auch Wildcards. Dabei gelten die Unix-Regeln. Das heißt, z. B. ``*'` ersetzt alle Dateien eines Verzeichnisses wie ``* . '*'` bei MS-DOS.

Bei der Benutzung von ``\`` oder Wildcards muß der Pfad durch einfache Quotes eingeschlossen werden, um ihn vor der Interpretation durch die Shell zu schützen.

Die Optionen werden im Unix-Stil von einem ``-'` eingeleitet, nicht von einem ``/'` wie bei MS-DOS.

In der Datei ``~/mcwd'` im Heimatverzeichnis des Benutzers ist das aktuelle MS-DOS Verzeichnis für alle `mttools` festgelegt. Dieses Verzeichnis kann mit dem `mcd`-Kommando gewechselt werden.

## Autor:

Emmet P. Gray



**Next:** [my](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [mt](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

**Next:** [newgrp](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [mtools](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- 

# mv

## Funktion:

**mv** (move) verschiebt eine Datei oder benennt sie um

## Syntax:

**mv** [*Optionen*] *Quelle* *Ziel*

**mv** [*Optionen*] *Quelle* ... *Verzeichnis*

## Beschreibung:

**mv** verschiebt eine oder mehrere Datei(en) bzw. Verzeichnis(se) oder benennt sie um. Ein Verzeichnis kann nicht über die Grenzen eines Dateisystems hinweg verschoben werden.

## Optionen:

-b  
sichert Dateien vor dem Überschreiben

-f  
überschreibt existierenden Zielf Dateien rücksichtslos

-i  
erwartet interaktiv eine Bestätigung vor dem Überschreiben existierender Zielf Dateien

-u  
verschiebt Dateien nur, wenn sie neuer sind als die gleichnamigen Zielf Dateien

-v  
meldet jede Aktion

-S *Endung*

bestimmt die *Endung* für einfaches Backup; Voreinstellung ist `~'

-V {numbered, existing, simple}

kontrolliert die Art des Backups; die Parameter sind beim Befehl [cp](#) erklärt

## Siehe auch:

[ln](#) und [cp](#)

## Autor:

Mike Parker und David MacKenzie

---

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

Next: [nice](#) Up: [Von GNU's, Muscheln und](#) Previous: [my](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
- 

# newgrp

## Funktion:

**newgrp** ändert die Gruppenkennung des aktuellen Benutzers

## Syntax:

**newgrp** [*Gruppe*]

## Beschreibung:

Das **newgrp**-Kommando ändert die aktive Gruppenkennung des Anwenders. Ohne Angabe einer Gruppe wird zu der in der `/etc/passwd` Datei festgelegten Standardgruppe des Benutzers gewechselt. Sonst wird in die angegebene Gruppe gewechselt, wenn der Benutzer in der Datei `/etc/group` als Mitglied dieser Gruppe eingetragen ist. Das **newgrp**-Kommando unterstützt keine paßwortgeschützten Gruppen.

Die Gruppe kann nur mit ihrem Namen angegeben werden.

## Autor:

Michael Haardt, Peter Orbaek

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- 

# nice

## Funktion:

**nice** läßt ein Programm mit veränderter Priorität laufen

## Syntax:

**nice** [-n *Nettigkeit*] [-*Nettigkeit*] [-adjustment=*Nettigkeit*] [*Kommando* [*Argument* ...]]

## Beschreibung:

In einem Multitasking Betriebssystem wie Linux muß die Prozessorzeit auf verschiedene Prozesse verteilt werden. Dazu gibt es einen 'Scheduler' der dafür sorgt, daß die Prozessorzeit möglichst optimal zugeteilt wird. Prozesse, die auf das Ergebnis eines anderen Prozesses, ein Interrupt oder ein Signal warten, können beispielsweise so lange 'schlafen', bis das erwartete Ereignis eingetreten ist. Der Kernel weckt die schlafenden Prozesse dann auf und teilt ihnen wieder Prozessorzeit zu. Trotzdem gibt es natürlich in der Regel mehr als einen lauffähigen Prozeß. Und der Scheduler muß nach bestimmten Regeln den lauffähigen Prozessen Rechenzeit zuteilen. Dabei benutzt er unter anderem den **nice**-Wert. Will ein Anwender **nice**, das heißt nett, zu den anderen Benutzern des Systems sein, startet er die Prozesse, die ruhig etwas länger dauern dürfen mit, dem **nice**-Kommando. Ein negatives **nice** ist nur der Superuserin erlaubt.

Wenn kein Wert angegeben ist, wird die **Nettigkeit** um 10 Punkte erhöht. Die maximale **Nettigkeit** ist 19 Punkte. Nach unten kann die Superuserin ihre **Nettigkeit** bis -20 abkühlen.

## Optionen:

-n *Wert*

-*Wert*

setzt **nice** auf *Wert*

## Siehe auch:

[nohup](#)

## Autor:

David MacKenzie

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [nl](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [newgrp](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

**Next:** [nohup](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [nice](#)

## Subsections

- [Funktion](#)
  - [Syntax](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- 

# nl

## Funktion

**nl** numeriert die Zeilen in einer Datei

## Syntax

```
nl [-h Stil] [-b Stil] [-f Stil] [-p] [-d zwei Zeichen] [-v Nummer]
[-i Nummer] [-l Nummer] [-s Zeichenkette] [-w Nummer] [-n
{ln,rn,rz}] [-header-numbering=Stil] [-body-numbering=Stil]
[-footer-numbering=Stil] [-first-page=Nummer]
[-page-increment=Nummer] [-no-renumber] [-join-blank-lines=Nummer]
[-number-separator=Zeichenkette] [-number-width=Nummer]
[-number-format={ln,rn,rz}] [-section-delimiter=zwei Zeichen] [Datei
...]
```

## Beschreibung:

**nl** gibt die Zeilen einer oder mehrerer Dateien (oder der Standardeingabe) mit Zeilennummern auf die Standardausgabe. Es können dabei die Zeilen einer (logischen) Seite in einen Kopf, einen Körper und einen Fuß unterteilt werden, die jeweils einzeln und in unterschiedlichen Stilen numeriert werden. Jeder Teil kann auch leer sein. Wenn vor dem ersten Kopfteil bereits Zeilen vorhanden sind, werden diese Zeilen wie ein Seitenkörper numeriert.

Die Numerierung beginnt auf jeder Seite neu. Mehrere Dateien werden als ein einziges Dokument betrachtet und die Zeilennummer wird nicht zurückgesetzt.

Der Kopfteil wird durch eine Zeile eingeleitet, die nur die Zeichenkette `\\:\\:' enthält. Der Körper wird entsprechend durch `\\:\\:' und der Fuß durch `\\:' eingeleitet. In der Ausgabe werden diese Zeilen als Leerzeilen ausgegeben.

# Optionen:

## -h *Stil*

bestimmt die Art der Zeilennumerierung für die Kopfzeile; das Nummerntrennzeichen wird auch den nicht numerierten Zeilen vorangestellt; als *Stil* werden folgende Zeichen erkannt

a

alle Zeilen werden numeriert

t

die leeren Zeilen werden nicht numeriert (Voreinstellung für den Körper)

n

die Zeilen werden nicht numeriert (Voreinstellung für Kopf und Fuß)

## p *Ausdruck*

nur die Zeilen, in denen der reguläre *Ausdruck* vorkommt, werden numeriert

## -b *Stil*

bestimmt die Art der Zeilennumerierung für den Körper

## -f *Stil*

bestimmt die Art der Zeilennumerierung für den Fuß

## -p

die Zeilen aller Seiten werden fortlaufend numeriert

## -v *Nummer*

die erste Zeile jeder logischen Seite bekommt die angegebene Nummer

## -i *Nummer*

die Schrittweite für die Numerierung

## -l *Nummer*

die angegebene Anzahl aufeinanderfolgender Leerzeilen werden als eine Zeile angesehen, und die letzte Zeile wird numeriert; wenn weniger Leerzeilen in Folge auftreten, werden sie nicht numeriert; Leerzeilen dürfen auch keine Leerzeichen oder Tabulatoren enthalten

## -s *Zeichenkette*

setzt die *Zeichenkette* als Nummerntrennzeichen zwischen Zeilennummer und Text; Voreinstellung ist `TAB`

## -w *Nummer*

die Zeilennummern erhalten die angegebene Anzahl Stellen; Voreinstellung ist 6

## -n {ln, rn, rz}

die Zeilennummern werden in dem angegebenen Stil ausgegeben; dabei bedeutet

ln

linksbündig, ohne führende Nullen

rn

rechtsbündig, ohne führende Nullen

rz

rechtsbündig, mit Nullen auf die volle Stellenzahl aufgefüllt



-d *zwei Zeichen*

die zwei Zeichen werden zur Trennung von Kopf, Körper und Fußteil benutzt, Voreinstellung ist `\'

## Siehe auch:

[pr](#)

### Autor:

Scott Bartram, David MacKenzie

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [nohup](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [nice](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
- 

# nohup

## Funktion:

**nohup** läßt ein Programm die Signale SIGHUP SIGINT SIGQUIT und SIGTERM ignorieren

## Syntax:

**nohup** *Kommando* [*Argument* ...]

## Beschreibung:

**nohup** schützt ein Programm vor den HANGUP-Signalen. Dadurch kann es im Hintergrund weiterlaufen, auch wenn der Benutzer sich ausloggt. Normalerweise würden mit der Loginshell alle Prozesse des Anwenders durch ein SIGHUP beendet.

Der Prozeß geht nicht automatisch in den Hintergrund, sondern muß mit einem ``&'` am Ende der Kommandozeile dorthin gebracht werden. Die Schedulerpriorität eines mit **nohup** gestarteten Programms wird um 5 erhöht. Wenn die Standardausgabe des Programms ein Terminal ist, so wird sie automatisch gemeinsam mit der Standardfehlerausgabe in die Datei `nohup.out` umgeleitet. Ist das aktuelle Verzeichnis schreibgeschützt, wird die Datei im HOME-Verzeichnis angelegt.

Next: [passwd](#) Up: [Von GNU's, Muscheln und](#) Previous: [nohup](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# od

## Funktion:

**od** (octal dump) zeigt Dateien im oktalen, hexadezimalen oder in anderen Formaten an.

## Syntax:

```
od [-abcdfhiloxv] [-s [Länge]] [-w [Anzahl]] [-A Positionsformat] [-j  
Anzahl] [-N Anzahl] [-t Format] [-skip-bytes=Anzahl]  
[-address-radix=Positionsformat]  
[-read-bytes=Anzahl] [-format=Format] [-output-duplicates]  
[-strings[=Anzahl]] [-width[=Anzahl]] [Datei ...]
```

## Beschreibung:

**od** liest die angegebenen Dateien oder die Standardeingabe (wenn keine Datei oder anstelle einer Datei '-' angegeben ist) und gibt die Bytes formatiert und kodiert auf die Standardausgabe.

Jede Ausgabezeile enthält in der ersten Spalte die Positionsnummer des ersten in der Zeile dargestellten Bytes (vom Dateianfang gezählt). In den darauffolgenden Spalten werden die Daten aus der Datei in einem durch die Optionen kontrollierten Format angezeigt.

In der Standardeinstellung ohne Optionen gibt **od** die Position als 7-stellige Oktalzahl  und die Daten in 8 Spalten zu je zwei Bytes in oktaler Darstellung aus.

## Optionen:

### -A *Positionsformat*

zeigt die Position des ersten in einer Zeile dargestellten Bytes im Positionsformat; die folgenden Formate stehen zur Auswahl:

-d

siebenstellige Dezimalzahl 

-o

siebenstellige Oktalzahl  (Voreinstellung)

-x

sechsstellige Hexadezimalzahl 

-n

keine Positionsangabe

-j **Anzahl**

überspringt die ersten Anzahl Bytes der Datei und beginnt erst danach mit der Ausgabe; wenn die Zahl mit `0x' oder `0X' beginnt, wird sie als Hexadezimalzahl interpretiert, beginnt sie mit einer Null, wird sie als Oktalzahl behandelt, sonst als Dezimalzahl; der Zahl kann einer der Buchstaben b (blocks=512), k (kilo=1024) oder m (mega=1048576) folgen, die die Anzahl mit den entsprechenden Einheiten multiplizieren

-N **Anzahl**

gibt nur die angegebene Anzahl Bytes von der Datei aus; die Anzahl kann wie bei der `-j' Option von einer Einheit gefolgt werden

-t **Format**

wählt die Codierung für die Datenausgabe; wenn die -t Option mehrfach benutzt wird oder mehrere Formate gleichzeitig angegeben werden, gibt es für jedes Format jeweils eine entsprechende Zeile aus; folgende Formate werden unterstützt

a

(ascii) setzt das achte Datenbit aller Zeichen auf null, die druckbaren ASCII Zeichen werden als solche ausgegeben, und die nichtdruckbaren Steuerzeichen werden mit ihren in der ASCII Tabelle verwendeten ``Namen" bezeichnet; so wird das Zeilenende als CR bezeichnet, ein horizontaler Tabulator mit TAB und so weiter

c

(character) gibt die druckbaren ASCII Zeichen als solche aus, die nichtdruckbaren Zeichen werden, sofern möglich, als Backslashsequenz ausgegeben; \f ist hier z. B. ein Zeilenende, \t ein Tabulator und so weiter; Bytes die nicht druckbar sind und auch nicht als Backslashsequenz ausgegeben werden können, werden als Oktalzahl dargestellt

d

(decimal) gibt die Daten als vorzeichenbehaftete Dezimalzahlen aus; voreingestellt sind vier Bytes je Dezimalzahl

f

(float) gibt die Daten als Fließkommazahlen aus; voreingestellt sind acht Bytes je Fließkommazahl

o

(octal) gibt die Daten als Oktalzahlen aus; voreingestellt sind vier Bytes je Oktalzahl

u

(unsigned) gibt die Daten als vorzeichenlose Dezimalzahl aus; voreingestellt sind vier Bytes je Dezimalzahl

x

(heX) gibt die Daten als Hexadezimalzahlen aus; voreingestellt sind vier Bytes je Hexadezimalzahl

Außer die Typen a und c, die immer einzelne Bytes anzeigen, kann die Anzahl der Bytes, die in jeweils eine Zahl des bestimmten Typs umgewandelt werden sollen, durch eine dem Typkennzeichner unmittelbar folgende Zahl bestimmt werden. Die Anzahl der Bytes je Zahl kann auch durch Buchstabenkennungen angegeben werden, die den C-Datentypen entsprechen. Für die Ganzzahltypen (d, u, x, o) gibt es die folgenden Möglichkeiten:

C

(Char) ist ein Byte lang

S

(Short) ist zwei Bytes lang

I

(Integer) ist vier Bytes lang

L

(Long) ist auch vier Bytes lang

für Fließkommazahlen können die folgenden Optionen verwendet werden

F

(Float) ist vier Bytes lang

D

(Double) ist acht Bytes lang

L

(Long Double) ist auch acht Bytes lang

-v

gibt auch die doppelten Zeilen aus; normalerweise werden ganze Zeilen, die der zuletzt angezeigten Zeile vollständig entsprechen, durch ein Asterisk '\*' dargestellt

-s [*Länge*]

gibt nur die gültigen C-Zeichenketten (Folgen von druckbaren ASCII Zeichen, durch ein Nullbyte beendet) mit mindestens der angegebenen *Länge* aus; Voreinstellung für Länge ist drei Zeichen

-w [*Anzahl*]

setzt die Anzahl der umgewandelten Bytes, die in einer Zeile ausgegeben werden; die Anzahl muß ein vielfaches der Länge jedes Ausgabetyps sein; Voreinstellung ist 16

Die folgenden Optionen übersetzen die nicht dem POSIX Format entsprechenden alten Optionen in die neuen, dem POSIX Standard entsprechenden Optionen, wie sie oben aufgeführt sind.

-a

gibt benannte ASCII Zeichen aus, wie -t a

-b

gibt die Daten Byteweise als Oktalzahlen aus, wie -t oC

-c

gibt druckbare Zeichen oder Backslashsequenzen aus, wie -t c

-d

- t u2 gibt die Daten als vorzeichenlose kurze Dezimalzahlen aus, wie -t u2
- f gibt die Daten als Fließkommazahlen mit vier Bytes je Zahl aus, wie -t fF
- h gibt die Daten als vierstellige Hexadezimalzahl aus, wie -t xL
- i gibt die Daten als vorzeichenbehaftete Dezimalzahl mit zwei Bytes je Zahl aus, wie -t d2
- l gibt die Daten als vorzeichenbehaftete Dezimalzahl mit vier Bytes je Zahl aus, wie -t dL
- o gibt die Daten als Oktalzahlen mit zwei Bytes je Zahl aus, wie -t oS
- x gibt die Daten als Hexadezimalzahlen mit zwei Bytes je Zahl aus, wie -t x2

**Autor:**

Jim Meyering

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [passwd](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [nohup](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

Next: [paste](#) Up: [Von GNU's, Muscheln und](#) Previous: [od](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Siehe auch:](#)
- 

# passwd

## Funktion:

**passwd** ändert das Paßwort zum System

## Syntax:

**passwd** [ *Username* ]

## Beschreibung:

Die Paßwörter aller Benutzer werden in der Datei `/etc/passwd` gespeichert. Diese Datei ist lesbar aber schreibgeschützt. Um dem Benutzer die Möglichkeit zu geben, sein eigenes Paßwort zu ändern, läuft **passwd** SUID **root**. Deshalb hat der Anwender zur Laufzeit des Programms Rootprivilegien und darf in die Datei schreiben.

Bei einigen Linux-Installationen wird das Benutzerpaßwort in einer separaten Datei namens `shadow` gespeichert, um den normalen Benutzern den Lesezugriff auf diese Daten zu verwehren. Die Einzelheiten zu diesem Paßwortsystem sind in den englischen Manualpages beschrieben.

## Siehe auch:

[chsh](#) und [newgrp](#)

## Autor:

Peter Orbaek

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# paste

## Funktion:

**paste** fügt die Zeilen von zwei oder mehr Dateien horizontal zusammen

## Syntax:

```
paste [-s] [-d Liste] [-serial] [-delimiters=Liste] [Datei ...]
```

## Beschreibung:

**paste** fügt die Zeilen mehrerer Dateien zusammen. Die Zeilen werden standardmäßig durch `TAB` getrennt, und die Ausgabe einer kompletten Zeile (das heißt die Ausgabe der entsprechenden Zeilen aller Dateien) wird mit einem Zeilenende abgeschlossen.

## Optionen:

`-d Liste`

benutzt die Zeichen aus *Liste* zur Trennung der Zeilen aus den einzelnen Dateien beim Zusammenfügen; *Liste* ist dabei ein Wort oder eine Zeichenkette aus beliebigen druckbaren Zeichen oder den Sonderzeichen `\n` `\t` `\\` und `\0` für Zeilenende, Tabulator, Backslash oder Leerstring; wenn die *Liste* abgearbeitet ist, wird sie von vorne angefangen

`-s`

fügt alle Zeilen einer ganzen Datei zu einer Zeile zusammen; werden mehrere Dateien angegeben, so werden die Zeilen der nächsten Dateien als jeweils eine neue Zeile angefügt

## Autor:

David M. Ihnat





**Next:** [printenv](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [paste](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# pr

## Funktion:

**pr** formatiert Textdateien zur Druckerausgabe

## Syntax:

```
pr [+SEITE] [-SPALTEN] [-abcdfFmrtv] [-e[Eintab[Schritt]]] [-h Kopf]
[-i[Austab[Schritt]]] [-l Seitenlänge] [-n[Separator[Stellen]]] [-o
Lrand] [-s[Separator]] [-w Breite] [Datei ...]
```

## Beschreibung:

**pr** produziert seitenweise numerierte und einfach formatierte Ausgabe von Textdateien.

## Optionen:

**+Seite**

beginnt die Ausgabe mit der angegebenen *Seite*

**-Spalten**

setzt den Text in die angegebene Anzahl *Spalten*; ein Zeilenumbruch findet nicht statt; die Spaltenbreite wird an die Seitenbreite angepaßt

**-a**

teilt den Text zeilenweise in die Spalten anstatt seitenweise

**-b**

schließt die letzte Seite mit balancierten Spalten ab, das heißt alle Spalten sind gleichlang

**-c**

zeigt Controlzeichen als Caretsequenz (^ ^G' für CONTROL-G)

**-d**

gibt den Text mit doppeltem Zeilenabstand aus

–e ***Eintab Schritt***

übersetzt das Zeichen *Eintab* in *Schritt* Leerzeichen; Voreinstellungen sind `TAB` für *Eintab* und 8 für *Schritt*

–f

gibt einen Seitenvorschub anstelle von Zeilenvorschüben am Seitenende

–h ***Kopf***

schreibt die Zeichenkette *Kopf* anstelle des Dateinamen als Seitenkopf

–i ***Austab Schritt***

ersetzt Folgen von *Schritt* Leerzeichen durch ein *Austab* Zeichen; voreingestellt sind `TAB` für *Austab* und 8 für *Schritt*

–l ***Seitenlänge***

setzt die Druckseitenlänge; Voreinstellung ist 66; bei einer Seitenlänge unter 10 Zeilen werden die Kopf- und Fußzeilen weggelassen

–m

gibt alle Dateien parallel in Spalten aus

–n ***Separator Stellen***

setzt Zeilennummern vor jede Spalte; werden Dateien parallel angezeigt, bekommt jede Zeile nur eine Nummer; der *Separator* steht zwischen der Nummer und der Zeile, Voreinstellung ist `TAB`; die Zeilennummer hat die angegebene Anzahl *Stellen*, Voreinstellung ist 5

–o ***Lrand***

setzt den linken Rand auf *Lrand*; die Seitenbreite ist die Summe von *Lrand* und *Breite* von Option –w

–r

gibt keine Fehlermeldung für Dateilesefehler aus

–s ***Separator***

trennt die Spalten durch den Buchstaben *Separator*; Voreinstellung ist `TAB`

–t

der 5-zeilige Kopf und der 5-zeilige Fußbereich werden nicht ausgegeben und die Seiten werden nicht durch Leerzeilen oder mit einem Seitenvorschub aufgefüllt

–v

gibt nichtdruckbare Sonderzeichen als Oktalzahl aus

–w ***Breite***

setzt die druckbare *Breite*; Voreinstellung ist 72 Zeichen

## Autor:

Pete TerMaat

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [printenv](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [paste](#)



## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
- 

# printenv

## Funktion:

**printenv** zeigt die Umgebungsvariablen für Programme

## Syntax:

**printenv** [*Variable* ...]

## Beschreibung:

**printenv** zeigt alle oder einzelne Umgebungsvariablen für Programme. Werden *Variable* in der Kommandozeile übergeben, so gibt der Status Null an, daß alle Variablen in der Umgebung gesetzt sind; sonst ist der Status Eins.

## Autor:

David MacKenzie und Richard Mlynarik

**Next:** [pwd](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [printenv](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# ps

## Funktion:

**ps** (process status) zeigt die Prozesse mit ihrem Status an


## Syntax:

**ps** -acehjlmnrsuvwxS -txx *Systempfad Swappfad*

## Beschreibung:

Mit **ps** lassen sich Daten über die Prozesse in der Prozeßtafel anzeigen.

Es gibt zwei unterschiedliche Versionen von **ps**. Das eine **ps** greift direkt auf den Kernelspeicher zu, aus dem es die Prozeßtafel und andere Daten ausliest. Dazu braucht **ps** die Datei `/etc/psdatabase`, in der die Speicheradressen für die entsprechenden Kernelvariablen abgelegt sind. Diese Datei muß für jeden Kernel mit dem Kommando ``ps -U'` neu erzeugt werden. Bei größeren Veränderungen am Kernel (in der Regel bei neuen Kernelversionen) wird auch ein Neuübersetzen von **ps** notwendig.

Das andere **ps** hat die gleiche Funktionalität und mit Ausnahme der `-U` Option auch die gleichen Optionen, es arbeitet aber mit dem Prozeßdateisystem. Dieses Dateisystem enthält Verzeichnisse  für alle Prozesse des Systems, in deren Unterverzeichnissen und Dateien alle für **ps** relevanten Daten zu finden sind. Das **ps**-Kommando bereitet diese Daten auf und zeigt sie dem Anwender in der gleichen Weise an wie die andere Version. Der Vorteil der Methode mit dem Prozeßdateisystem besteht in der Unabhängigkeit von der Kernelversion.

Die beiden Versionen von **ps** unterscheiden sich in verschiedenen Details bei den Ausgabeformaten und in den Kommandozeilenoptionen. Die Beschreibung hier bezieht sich hauptsächlich auf das `procps`. Die Optionen und Ausgabeformate des kernelabhängigen **ps** werden nur am Rande und unvollständig dokumentiert.

Die Prozeßtafel wird mit einer Titelzeile ausgegeben. Die Spalten bedeuten folgendes:

COMMAND

der Name des Kommandos; Prozesse, die komplett in den Swapbereich ausgelagert sind, werden in Klammern angezeigt

TIME

die verbrauchte Rechenzeit (Summe User- und Kernelmodus) im Format MM:SS

TT

die Nummer des kontrollierenden Terminals

UID

die Benutzer-ID des Eigentümers dieses Prozesses

PID

die Prozeßnummer dieses Prozesses

PPID

die Prozeßnummer des Elternprozesses

PGID

die Prozeßgruppe dieses Prozesses

TPGID

die Prozeßgruppe, der z. Z. das kontrollierende Terminal zu diesem Prozeß gehört

SID

die Session-ID des Prozesses (ID der Loginshell)

STAT

ist der Status des Prozesses; folgende Symbole sind möglich:

R

lauffähig

S

schlafend

D

nicht störbarer Schlaf

T

angehalten oder verfolgt

Z

Zombie

Die folgenden Statusinformationen werden vom `procps` nicht angeboten.

W

der Prozeß belegt keine Seiten im Arbeitsspeicher

I

(idle) der Prozeß läuft leer

P

der Prozeß wird gerade in den Swapbereich ausgelagert

x

der Prozeß wird mit dem Debugger verfolgt

X

die System-Calls werden verfolgt

S

der Prozeß führt eine neue Session an

+

der Prozeß ist in einer Prozeßgruppe im Vordergrund

<

kennzeichnet die Prozesse mit höherer Priorität

N

kennzeichnet die Prozesse mit verminderter Priorität

F

die Flags des Prozesses

10

der Prozeß wird verfolgt (traced); das ist z. B. im Debugger der Fall

20

die System-Calls des Prozesses werden verfolgt

40

der Prozeß hat sich selbst von seinem Elternprozeß gelöst (häufig bei Dämonen)

100

der Prozeß läuft mit Rootrechten

200

der Prozeß hat einen Coredump ausgelöst

400

der Prozeß wird durch ein Signal beendet

100000

Prozeß hat die FPU (den mathematischen Coprozessor) benutzt

PRI

das aktuelle Maximum an Rechenzeit (in Millisekunden), das dem Prozeß zugeteilt wird; wenn der Prozeß gerade läuft, ist das der Rest von der Zeitscheibe

NI

ist der Nicewert des Prozesses; dieser Wert kann die Geschwindigkeit erhöhen oder verringern, in der die Zeitscheibe eines Prozesses verbraucht wird

MAJFLT

(auch **PAGEIN**) Anzahl der „major page faults“ (das sind die Versuche, auf eine ausgelagerte Seite zuzugreifen)

MINFLT

Anzahl der „minor page faults“; diese Zugriffe hatten keine Hardwareaktion zur Folge

TSIZ

(Textsize) die Größe des Textsegmentes der ausführbaren Datei

DSIZ



(Datasize) die Differenz aus `vsize` (virtuelle Größe des Prozesses) und `TSIZ`

## RSS

(Resident Set Size) ist die Größe des Programms im Arbeitsspeicher; dieser Wert wird aus dem `task_struct` errechnet und ist nicht mit dem RSS Feld in `statm` identisch

## SIZE

der „virtuelle“ Speicherbereich des Prozesses

## STACK

zeigt auf die Basis des nach unten wachsenden Stack

## LIM

zeigt den mit `ulimit -m` eingestellten Grenzwert für den Resident Stack Size

## %MEM

## ESP

(Extended Stack Pointer) zeigt auf die Spitze des nach unten wachsenden Stack

## EIP

(Extended Instruction Pointer) zeigt auf den aktuellen Maschinenbefehl im Programmtext (<0x60000000) oder in den Shared Libraries (>0x60000000)

## TMOUT

zeigt den Wert eines eventuell gesetzten Timeouts

## ALARM

zeigt den Wert eines eventuell gesetzten Alarmtimers (z. B. von `sleep`)

## SIGNAL

zeigt ein eventuell gerade anliegendes Signal

## BLOCKED

Bitmaske der blockierten Signale

## IGNORED

Bitmaske der Signale, die ignoriert werden

## CATCHED

Bitmaske für Signale, die abgefangen werden

## WCHAN

ist der Name der Kernelfunktion, in der der Prozeß schläft 

Die folgenden Werte werden mit der `-m` Option angezeigt. Sie geben detailliert Auskunft über die Speicherauslastung durch die einzelnen Prozesse. Die Werte für `SIZE` und `RSS` werden auch mit anderen Optionen angezeigt, die mit der `-m` Option ausgegebenen Werte werden auf eine andere Weise berechnet.

## TRS

(Text Resident Size) Größe des Textsegments (enthält keine shared Libraries)

## DRS

(Data Resident Size, auch **DSIZ**) Größe des Datensegments (enthält benutzte Libraryseiten)

## SIZE

die Speicherbelegung des Prozesses; Text, Daten und Stack unabhängig davon, ob sie sich im physikalischen Speicher befinden

## SWAP

ausgelagerte Speicherseiten in Kilobyte (oder Seiten mit -p); ist einfach die Differenz aus SIZE und RSS

## RSS

(Resident Set Size)

## SHRD

die Größe des mit mindestens einem anderen Prozeß gemeinsam benutzten Speicherbereiches

## LIB

(Library Resident Size) die gesamte Größe der vom Prozeß benutzten Libraryseiten im Arbeitsspeicher

## DT

(Dirty) benutzte Libraryseiten in Kilobyte (oder Seiten mit -p)

## Optionen:

- a  
zeigt die Prozesse aller User
- c  
zeigt den Namen des Kommandos
- e  
zeigt die Prozeßumgebung
- h  
unterdrückt die Kopfzeile
- j  
jobs Format: PGID und SID
- l  
langes Format: FLAGS WCHAN NICE PRIO
- m  
zeigt Speichernutzung
- X  
zeigt EIP ESP TIMEOUT und ALARM
- n  
gibt numerische Werte für USER und WCHAN
- r  
zeigt nur die laufenden Prozesse
- s  
zeigt die Signale
- u

zeigt die Besitzer der Prozesse

-v

vm Format

-w

„breite“ Ausgabe, kann bis zu drei mal angegeben werden; die Ausgabezeilenlänge wird auf 132 Spalten, 264 Spalten und unbegrenzt erhöht

-x

zeigt Prozesse, die von keinem Terminal kontrolliert werden

-S

addiert die Prozessorzeit der Kindprozesse zu den Eltern

t *xx*

zeigt nur die Prozesse, die von Terminal *xx* kontrolliert werden

Folgende Optionen werden nur vom kernelabhängigen `ps` zusätzlich angeboten:

-0

der Prozeß Nummer 0 (scheduler) wird mit angezeigt

-f

(forest) zeigt die Prozeßfamilien mit ihren Eltern-Kind-Beziehungen als Bäume


-H

gibt einen kurzen Hilfstext zum `ps`-Kommando aus

-p

veranlaßt `ps` die Speicherbelegung in Seiten und nicht in Kilobytes anzuzueigen

-U

aktualisiert die Datei `/etc/psdatabase`, die den Zugang zu den Kerneldaten vermittelt; diese Aktualisierung muß immer durchgeführt werden, nachdem der Kernel neu übersetzt wurde; diese Option fällt bei dem `ps` Programm ,das mit dem Prozeßdateisystem arbeitet, weg 

-Y

zeigt den Syscall, in dem der Prozeß gerade schläft oder den er gerade verlassen hat

## Autor:

Branko Lankester, Michael K. Johnson, Rick Sladkey

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [pwd](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [printenv](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

**Next:** [rm](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [ps](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
- 

# pwd

## Funktion:

**pwd** gibt den Namen des aktuellen Verzeichnisses aus

## Syntax:

```
pwd
```

## Beschreibung:

**pwd** ist in vielen Shells ein integriertes Kommando (z. B. in der [bash](#)). Wenn in der Distribution kein binäres **pwd**-Kommando enthalten ist, kann als Notbehelf beispielsweise mit der `tcsh`, bei der kein **pwd**-Kommando integriert ist, das folgende Shellscrip benutzt werden:

```
#!/bin/bash
pwd
```

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# rm

## Funktion:

**rm** löscht Dateien

## Syntax:

```
rm [-dfirvR] [-directory] [-force] [-interactive] [-recursive]  
[-verbose] Pfad ...
```

## Beschreibung:

**rm** löscht Dateien. Normalerweise werden die Verzeichnisse nicht mitgelöscht. Wenn eine Datei gelöscht werden soll, für die keine Schreibberechtigung besteht, muß der Befehl für diese Datei extra bestätigt werden. In Verzeichnissen, bei denen das Stickybit gesetzt ist, kann eine Datei nur von ihrem Eigentümer gelöscht werden.

Die Option ``-'` zeigt an, daß die folgenden Argumente keine Optionen mehr sind. Dadurch ist es möglich, auch Dateinamen, die mit einem ``-'` anfangen, zu löschen.

## Optionen:

`-d`

löscht Verzeichnisse mit dem ``unlink'` Systemaufruf anstelle von `rmdir` (nur für die Superuserin **Ruth**); weil die in einem so gelöschten Verzeichnis enthaltenen Dateien nicht mitgelöscht werden, ist eine anschließende Reparatur des Dateisystems angesagt

`-f`

keine Nachfragen, keine Fehlermeldungen

`-i`

vor dem Löschen jeder Datei wird nochmal nachgefragt

`-r`

der Inhalt aller Unterverzeichnisse und die Verzeichnisse werden mitgelöscht

-v

zeigt die Namen aller Dateien noch ein letztes Mal an, bevor sie gelöscht werden

## Autor:

Paul Rubin, David MacKenzie und Richard Stallman

---

Next	Up	Previous	Contents	Index
------	----	----------	----------	-------

**Next:** [rmdir](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [pwd](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

**Next:** [sed](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [rm](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# rmmdir

## Funktion:

**rmmdir** löscht Verzeichnisse

## Syntax:

```
rmmdir [-p] [-path] Verzeichnis ...
```

## Beschreibung:

**rmmdir** löscht leere Verzeichnisse. Es gibt keine Möglichkeit, Verzeichnisse zu löschen, die noch normale Dateien enthalten.

## Optionen:

**-p**

löscht mehrere Verzeichnisse rekursiv, wenn alle Verzeichnisse leer sind (nachdem das Verzeichnis im Verzeichnis gelöscht ist)

## Autor:

David MacKenzie

**Next:** [setfdprm](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [rmdir](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# sed

## Funktion:

**sed** (stream editor) ist ein Editor zur nicht-interaktiven Textbearbeitung

## Syntax:

```
sed [-nV] [-quiet] [-silent] [-version] [-e Editorkommando] [-f Scriptdatei] [-expression=Editorkommando] [-file=Scriptdatei] [Datei ...]
```

## Beschreibung:

**sed** ist ein Editor zur automatischen Textbearbeitung.

Die Bearbeitung erfolgt mit Editorkommandos, die dem **sed** in einer separaten *Scriptdatei* oder direkt in der Kommandozeile übergeben werden. Um in der Kommandozeile mehrere *Editorkommandos* zu übergeben, kann die `-e` Option mehrfach verwendet werden. Die Editorkommandos können auch durch ein Semikolon getrennt werden. Wird nur ein einziges Editorkommando in der Kommandozeile übergeben, kann die Kennzeichnung mit der `-e` Option auch weggelassen werden. Damit die Shell keine Veränderungen an der Zeichenkette mit dem Editorkommando vornimmt, muß sie in Hochkommata eingeschlossen werden.

Eine Scriptdatei kann beliebig viele Editorkommandos enthalten, die durch Zeilenende oder Semikolon von einander getrennt werden müssen.

Jedes Kommando besteht aus einem Adreßteil und einem Funktionsteil. Der Adreßteil gibt an, welche Zeilen einer Textdatei mit diesem Kommando bearbeitet werden sollen, und der Funktionsteil beschreibt die Veränderung, die an den im Adreßteil bestimmten Zeilen vorgenommen werden soll. Wenn kein Adreßteil angegeben ist, wird die Funktion mit jeder Zeile ausgeführt.

Die Bearbeitung eines Textes erfolgt, indem die Eingabe zeilenweise in einen Arbeitsspeicher gelesen wird und dann die Adreßteile aller Editorkommandos der Reihe nach mit dem Text im Arbeitsspeicher verglichen werden. Die Funktionen der passenden Kommandos werden in der



Reihenfolge ihres Auftretens ausgeführt. Normalerweise wird nach der Bearbeitung aller Kommandos der Inhalt des Arbeitsspeichers auf die Standardausgabe ausgegeben und danach durch die nächste Eingabezeile ersetzt. Die automatische Ausgabe des Arbeitsspeichers nach jeder Zeile kann mit der Option `-n` unterdrückt werden.

Zusätzlich zu dem Arbeitsspeicher gibt es noch einen Zwischenspeicher (Puffer), der von verschiedenen Funktionen benutzt werden kann.

Der Arbeitsspeicher kann auch mehrere Zeilen auf einmal enthalten.

Im Adreßteil können die Zeilen entweder durch ihre Zeilennummern oder durch reguläre Ausdrücke ausgewählt werden. Alle Funktionen außer den ``a'`, ``i'`, ``q'` und ``='` akzeptieren einen Adreßbereich, bei dem eine Start- und eine Endadresse durch ein Komma getrennt angegeben werden. Ein Dollarzeichen steht für die letzte Zeile. Wenn eine Endadresse mit einem regulären Ausdruck bezeichnet ist, wird die erste passende Zeile als Bereichsende eingesetzt.

Reguläre Ausdrücke müssen in einfachen Schrägstrichen (Slashes ``/'`) eingeschlossen werden. `sed` benutzt die gleichen Routinen zur Auswertung regulärer Ausdrücke wie `emacs` oder [grep](#)). Darüber hinaus kann auch die an die `ed` Syntax angelehnte Konstruktion ``\#Muster#'` (mit jedem beliebigen Zeichen für ``#'`) benutzt werden, die wie `/Muster/` interpretiert wird.

Im *Muster* kann auch ein ``\n'` vorkommen, das auf das Zeilenende paßt.

Der `sed` kann folgende Funktionen ausführen:

**a***Text*  
schreibt den *Text* in die Standardausgabe, bevor die nächste Eingabezeile gelesen wird

**b** *Marke*  
springt zur mit der `:Marke` markierten Zeile im Script (nicht im Text) und fährt dort mit dem Programm fort

**c***Text*  
die im Arbeitsspeicher von `sed` befindliche Zeilen werden gelöscht und der Text in die Standardausgabe geschrieben; wenn ein Adreßbereich angegeben ist, wird der Text erst am Bereichsende einmal ausgegeben

**d**  
alle aktuell im Arbeitsspeicher von `sed` befindlichen Zeichen werden gelöscht und die nächste Eingabezeile gelesen; die auf diesen Befehl folgenden Befehle werden nicht mehr bearbeitet, auch wenn die Zeilen im Arbeitsspeicher im passenden Bereich liegen

**D**  
die erste Zeile im Arbeitsspeicher von `sed` wird gelöscht und die nächste Zeile wird gelesen; die auf diesen Befehl folgenden Befehle werden nicht mehr bearbeitet, auch wenn die Zeilen im Arbeitsspeicher im passenden Bereich liegen

**g**  
der Arbeitsspeicher von `sed` wird durch den Inhalt des Puffers ersetzt; der Inhalt des Arbeitsspeichers geht dabei verloren

**G**  
der Pufferinhalt wird an den Inhalt des Arbeitsspeichers angehängt

**h**

der Inhalt des Arbeitsspeichers wird in den Puffer geschrieben; der Inhalt des Puffers geht dabei verloren

H

der Inhalt des Arbeitsspeichers von `sed` wird an den Puffer angehängt

i\Text

(insert) der Text wird sofort in die Standardausgabe geschrieben

l

der Inhalt des Arbeitsspeichers von `sed` wird ausgegeben; nichtdruckbare Zeichen werden als Oktalzahl dargestellt

n

der Inhalt des Arbeitsspeichers wird unverändert in die Ausgabe geschrieben und der Arbeitsspeicher durch die nächste Eingabezeile ersetzt

N

die nächste Eingabezeile wird an den Arbeitsspeicher angehängt; das Zeilenende wird mit in den Arbeitsspeicher geschrieben; die Zeilennummer des aktuellen Bereiches erhöht sich um eins

p

der Inhalt des Arbeitsspeichers wird in die Standardausgabe geschrieben

P

die erste Zeile im Arbeitsspeicher wird in die Standardausgabe geschrieben

q

beendet `sed`; es werden keine weiteren Befehle ausgeführt und keine Zeilen mehr gelesen

r *Datei*

der Inhalt der Datei wird ausgegeben, bevor die nächste Zeile gelesen wird

s/*Ausdruck/Ersetzung*[/*Modus*]

(substitute) ersetzt den (ersten) auf den regulären Ausdruck passenden Text durch den Ersetzungstext; es kann auch ein beliebiges anderes Zeichen anstelle von ``/'` benutzt werden; als Modus können ein oder mehrere der folgenden angegeben werden

**n**

eine Zahl von 1 bis 512 ersetzt nur das n-te Auftreten des Musters

g

(global) alle auf den Ausdruck passenden Textteile werden ersetzt

p

wenn eine Ersetzung stattgefunden hat, wird der Inhalt des Arbeitsspeichers von `sed` in die Standardausgabe geschrieben

w *Datei*

wenn eine Ersetzung stattgefunden hat, wird der Inhalt des Arbeitsspeichers in die *Datei* geschrieben

t *Marke*

verzweigt zur mit der *:Marke* versehenen Zeile in der Programmdatei, wenn eine Ersetzung am Inhalt des Arbeitsspeichers vorgenommen wurde, seit die letzte Eingabezeile gelesen wurde,

oder seit der letzte *t* Befehl bearbeitet wurde; wenn keine *Marke* angegeben ist, wird an das Ende der Programmdatei verzweigt

**w *Datei***

schreibt den Inhalt des Arbeitsspeichers in die benannte *Datei*

**x**

vertauscht den Inhalt des Puffers mit dem Arbeitsspeicher

**y/*Zeichenkette1*/*Zeichenkette2*/**

vertauscht jedes auftretende Zeichen aus der *Zeichenkette1* mit dem entsprechenden Zeichen der *Zeichenkette2*; die beiden Zeichenketten müssen gleich lang sein

**!*Funktion***

führt die Funktion für alle Zeilen aus, die NICHT in den Bereich passen

**:*Marke***

setzt eine *Marke* für den *b* und den *t* Befehl

**{...}**

die von den Klammern eingeschlossenen und durch Zeilenende oder Semikolon getrennten Funktionen werden als Einheit behandelt

**=**

gibt die aktuelle Eingabezeilennummer aus

**#**

leitet einen Kommentar ein; alle folgenden Zeichen bis zum Zeilenende werden ignoriert

## Optionen:

**-n**

gibt nur die Zeilen aus, die explizit (durch die Anweisung *p*) ausgedruckt werden sollen

**-V**

gibt die Versionsnummer und eine Kurzhilfe aus

**-e *Zeichenkette***

wendet die Editorbefehle aus *Zeichenkette* auf den Text an

**-f *Datei***

liest die Editorbefehle aus der *Datei*

## Autor:

Unbekannt

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [setfdprm](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [rmdir](#)

Das Linux Anwenderhandbuch

(C) 1997 [LunetIX](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- 

# setfdprm

## Funktion:

**setfdprm** lädt den Floppycontroller mit Parametern für ein Diskettenformat

## Syntax:

```
setfdprm [-p] [-c] [-y] [-n] Gerätedatei [Formatname]  
[Formatparameter]
```

## Beschreibung:

**setfdprm** stellt den Floppycontroller auf ein neues Diskettenformat ein.

Neben den 31 im Kernel gespeicherten Standardformaten sind theoretisch eine vielzahl weiterer Diskettenformate möglich. Da diese Formate vom Kernel nicht automatisch erkannt werden können, müssen die entsprechenden Parameter mit dem **setfdprm**-Kommando in den Floppycontroller geladen werden.

Die Parameter werden durch 9 Zahlen angegeben:

```
size sectors heads tracks stretch gap rate spec1 fmt_gap
```

Die passenden Werte lassen sich praktisch nur unmittelbar nach dem Formatieren der Diskette durch das Kommando **getfdprm** ermitteln. Die von **getfdprm** ausgegebene Zahlenfolge kann nach einem Diskettenwechsel unverändert in die Kommandozeile von **setfdprm** übernommen werden. Einfacher ist es, die Zahlenfolge unter einem beliebigen Namen in der Date zu speichern und später nur noch diesen Namen anstelle der Zahlenfolge an das **setfdprm**-Kommando zu übergeben.

## Optionen:

-p *Gerätedatei Name*

-p *Gerätedatei Parameter*

-c

löscht die Formatparameter

-y

veranlaßt den Floppytreiber, bei der automatischen Erkennung das gefundene Format als Kernelnachricht anzuzeigen

-n

schaltet die Formatanzeige bei der automatischen Erkennung ab

## Siehe auch:

`getfdprm(1)` und </etc/fdprm>

### Autor:

Werner Almesberger

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [sleep](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [sed](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

**Next:** [sort](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [setfdprm](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# sleep

## Funktion:

**sleep** läßt die Zeit verstreichen

## Syntax:

**sleep** *Zeit* [*smhd*] ...

## Beschreibung:

**sleep** wartet, daß die *Zeit* verstreicht. Voreingestellt sind Sekunden, es können aber auch Minuten, Stunden oder Tage sein. Diese Funktion wird vor allem in Shellscripts zur vorübergehenden Anzeige von Informationen benutzt.

## Optionen:

s	Sekunden
m	Minuten
h	Stunden
d	Tage

## Autor:

Free Software Foundation

---



Next: [split](#) Up: [Von GNU's, Muscheln und](#) Previous: [sleep](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# sort

## Funktion:

**sort** sortiert die Zeilen einer Textdatei

## Syntax:

```
sort [-cmus] [-t Separator] [-o Ausgabedatei] [-bdfiMnr] [+POS1  
[-POS2]] [-k POS1[,POS2]] [Datei...]
```

## Beschreibung:

**sort** wird normalerweise zum Sortieren von Dateien verwendet. Es kann aber auch Dateien daraufhin überprüfen, ob sie sortiert sind, oder mehrere sortierte (oder unsortierte) Dateien zu einer sortierten zusammenfügen.

Dazu existieren drei Modi:

- der **check** Modus, der prüft, ob eine Datei bereits sortiert ist. Dieser Modus wird durch die Option `-c` eingeleitet
- der **merge** Modus, der mehrere vorsortierte Dateien zusammenfügt. Die Dateien so zusammenzufügen ist schneller, als sie komplett sortieren zu lassen. Dieser Modus wird durch die Option `-m` eingeleitet
- Der Standardmodus ist **sort**

Wenn Schlüsselfelder bezeichnet sind, vergleicht **sort** die Schlüsselfelder in der Reihenfolge ihrer Bezeichnung, bis ein Unterschied gefunden wurde oder keine weiteren Felder vorhanden sind.

Wenn eine der globalen Optionen `Mbdfinr` benutzt wird, und kein Schlüsselfeld angegeben ist, vergleicht **sort** die ganzen Zeilen.



## Optionen:

- b  
ignoriert führende Leerzeichen
- c  
stellt fest, ob die Datei(en) bereits sortiert ist/sind; wenn eine Datei nicht sortiert ist, wird eine Fehlermeldung ausgegeben und mit dem Status 1 abgebrochen
- d  
sortiert in alphabetischer Reihenfolge
- f  
unterscheidet nicht zwischen Groß- und Kleinschreibung
- i  
ignoriert alle nicht druckbaren Zeichen (außerhalb 040-126 ASCII)
- M  
sortiert die (amerikanischen) Monate jan feb mar ... dec in der korrekten Reihenfolge; führende Leerzeichen werden wie bei -b ignoriert
- m  
fügt bereits sortierte Dateien zeilenweise zusammen
- n  
sortiert Zeilen mit Zahlen; ignoriert führende Leerzeichen und behandelt '-' als Vorzeichen
- r  
sortiert in umgekehrter Reihenfolge
- o *Datei*  
schreibt in die *Datei* anstelle der Standardausgabe; wenn eine der Eingabedateien als Ausgabedatei bestimmt wird, legt `sort` erst eine Kopie der Eingabedatei an und sortiert dann in die Ausgabedatei
- t *Separator*  
benutzt *Separator* als Feldtrenner für die Suchschlüssel; Standard ist der Leerstring zwischen einem Nichtblank und einem Blank; Der Trenner ist nicht Teil eines der getrennten Felder
- u  
im merge Modus wird nur die erste von einer Reihe gleichwertiger Zeilen ausgegeben; im check Modus wird geprüft, ob nicht zwei Zeilen gleichwertig sind
- +*POS1* [ -*POS2*]  
bestimmt die Zeichen zwischen *POS1* und *POS2* zum Sortierschlüssel; wenn *POS2* fehlt, werden alle Zeichen bis zum Zeilenende zum Schlüssel; Positionen der Felder und Buchstaben zählen von 0
- k *POS1* [ -*POS2*]  
bestimmt die Zeichen zwischen *POS1* und *POS2* zum Sortierschlüssel; wird das Schlüsselfeld so spezifiziert, zählen die Felder und Buchstaben von 1

Eine Position hat die Form *feld.buchstabe*

## Autor:

[Next](#) [Up](#) [Previous](#) [Contents](#) [Index](#)

**Next:** [split](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [sleep](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# split

## Funktion:

**split** spaltet eine Datei in mehrere kleinere

## Syntax:

```
split [-Zeilen] [-l Zeilen] [-b Bytes[bkm]] [-C Bytes[bkm]]  
[-lines=Zeilen] [-bytes=Bytes[bkm]] [-line-bytes=Bytes[bkm]] [Datei  
[Prefix]]
```

## Beschreibung:

**split** teilt eine Datei in mehrere Teile. Wenn keine weiteren Optionen gegeben sind, wird die *Datei* in Teile zu je 1000 Zeilen aufgeteilt. Die Ausgabe erfolgt in Dateien mit der Endung *Prefix* oder **x**, wenn kein Prefix angegeben wird.

## Optionen:

**-Zeilen**

die Ausgabedateien sind *Zeilen* lang

**-l Zeilen**

die Ausgabedateien sind *Zeilen* lang

**-b Bytes [bkm]**

die Ausgabedateien sind *Bytes* lang; die optionale Endung setzt die Einheit auf

b

512 Byte Blöcke

k

1 Kilobyte (1024) Blöcke

m

1 Megabyte Blöcke

–C *Bytes* [bkm]

schreibt so viele Zeilen wie möglich in die Ausgabedatei, ohne *Bytes* zu überschreiten; ist eine Zeile länger als *Bytes*, wird die Zeile auf mehrere Dateien aufgeteilt, bis der Rest weniger als *Bytes* lang ist; die Optionen bkm werden benutzt wie bei –b

**Siehe auch:**

[csplit](#)

**Autor:**

tege@sics.se

---

Next	Up	Previous	Contents	Index
------	----	----------	----------	-------

**Next:** [strace](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [sort](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# strace

## Funktion:

**strace** verfolgt Systemcalls und Signale eines Prozesses

## Syntax:

```
strace [-d][-t][-f][-s Länge][-o Datei] {-p ProzessID | Kommando}
```

## Beschreibung:

**strace** schaltet sich zwischen einen Benutzerprozeß und den Kernel und protokolliert alle Aktivitäten an dieser Schnittstelle. Wenn der verfolgte Prozeß einen Systemaufruf macht, gibt **strace** den Namen, die Argumente des Aufrufs und den Rückgabewert dieses Systemaufrufs aus. Wenn der verfolgte Prozeß ein Signal erhält, gibt **strace** den Namen des Signals aus.

Falls ein Systemaufruf mit einem Fehler zurückkehrt, wird, wenn vorhanden, die dem Fehlerstatus zugeordnete Fehlermeldung angezeigt. Zeichenketten als Argumente eines Systemaufrufs werden nur bis zu einer bestimmten Länge ausgegeben. Standardmäßig ist dieser Wert auf 32 Zeichen eingestellt. Wenn eine Zeichenkette länger als der eingestellte Wert ist, werden die fehlenden Zeichen durch zwei Punkte angedeutet.

Es ist möglich, einen einzelnen Prozeß oder eine ganze Prozeßfamilie zu verfolgen. Wenn die von einem verfolgten Prozeß erzeugten Kindprozesse auch verfolgt werden, setzt die Protokollierung erst ein, nachdem der das Kind erzeugende Systemaufruf zurückkehrt und die ProzeßID des Kindes an den Elternprozeß zurückgibt. Zu diesem Zeitpunkt kann der Kindprozeß bereits Systemaufrufe gemacht haben, die dann nicht protokolliert sind.

Wenn **strace** zu einem bereits laufenden Prozeß hinzugeschaltet wird, können nur solche Kindprozesse verfolgt werden, die nach dem Hinzuschalten erzeugt werden.

Wenn Sie es nicht anders bestimmen, schreibt **strace** das Protokoll in den Standardfehlerkanal.

Sie können das Protokoll auch in eine Datei schreiben lassen. Wenn Sie die Kinder des verfolgten Prozesses auch verfolgen, werden die Protokolle für die Kindprozesse in separaten Dateien

gespeichert, deren Namen mit dem für den Stammprozeß gewählten übereinstimmen und auf die ProzeßID des jeweiligen Kindes enden.

## Optionen:

- d  
(debug) veranlaßt `strap`, zusätzliche Informationen über eigene Systemaufrufe und Signale auf den Standardfehlerkanal zu schreiben
- t  
(time) läßt am Anfang jeder Protokollzeile die Zeit (Std:Min:Sec) ausgeben
- f  
(follow) veranlaßt die Verfolgung von Kindprozessen des verfolgten Prozesses
- o *Datei*  
veranlaßt `strace`, das Protokoll in die *Datei* zu schreiben
- s *Länge*  
verändert die standardmäßig auf 32 Zeichen beschränkte *Länge* der Ausgabe von Zeichenketten als Argument eines Systemaufrufs
- p *ProzeßID*  
schaltet `strace` zum laufenden Prozeß mit der Prozeßnummer *ProzeßID* hinzu; ohne Rootprivilegien können Sie nur Ihre eigenen Prozesse verfolgen

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [tty](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [split](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

Next: [su](#) Up: [Von GNU's, Muscheln und](#) Previous: [strace](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Beispiel:](#)
- 

# stty

## Funktion:

**stty** setzt die Terminalparameter oder zeigt sie an

## Syntax:


```
stty [-ag] [-all] [-save] [Einstellungen ...]
```

## Beschreibung:

Wenn **stty** ohne Argumente aufgerufen wird, gibt es die Leitungsgeschwindigkeit und alle Parameter, die von der Einstellung `sane` abweichen, für das aktuelle Terminal aus.

Alle Anzeigen und Einstellungen beziehen sich auf die Standardeingabe. Um also ein anderes als das aktuelle Terminal zu überprüfen oder einzustellen, muß die Standardeingabe von dem entsprechenden Device umgelenkt werden.

Mit den folgenden Argumenten lassen sich die Eigenschaften des Terminals ändern. Wird einem Argument ein ``-'` vorangestellt, so wird die entsprechende Eigenschaft abgeschaltet bzw. in ihr Gegenteil verkehrt.

Um alle Einstellungen in ihrer vollen Bedeutung zu verstehen, muß man sich mit der Funktionsweise der asynchronen, zeichenorientierten Gerätetreiber im Kernel im allgemeinen und der seriellen Schnittstelle im speziellen befassen. Dieses Thema führt aber weit über den Rahmen dieses Buches hinaus. 

## Allgemeine Einstellungen:

`parenb`

schickt und erwartet ein Paritätsbit bei der Datenübertragung auf der seriellen Schnittstelle

`parodd`

setzt ungerade Parität (gerade mit `-parodd`)

`cs5 cs6 cs7 cs8`

setzt die Zeichengröße auf 5, 6, 7 oder 8 Bits

`hupcl` | `hup`

wenn der letzte Prozeß die Gerätedatei schließt, werden DTR und RTS zurückgesetzt und dadurch ein Modem zum Unterbrechen einer eventuell noch bestehenden Telefonverbindung veranlaßt (`hangup on close`)

`cstopb`

jedes auf der seriellen Schnittstelle übertragene Zeichen wird durch zwei Stopbits abgeschlossen (ein Stopbit mit `-cstopb`)

`cread`

sollte den „Receiver“ der seriellen Schnittstelle abschalten; wird von Linux ignoriert

`clocal`

schaltet den „soft carrier“ für die serielle Schnittstelle ein, das CD Modemkontrollsignal wird vorgetäuscht; serielle Terminals brauchen normalerweise dieses Flag, auf Modemleitungen muß dagegen der „hard carrier“ eingeschaltet sein, damit das Modem die Schnittstelle kontrollieren kann

`crtsets`

schaltet RTS/CTS Hardware-Handshaking zur Datenflußkontrolle auf der seriellen Schnittstelle ein

## Einstellungen für die Terminaleingabe:

`ignbrk`

der Terminaltreiber ignoriert BREAK Zeichen

`brkint`

der serielle Treiber erzeugt ein SIGINT wenn vom Modem ein BREAK gesendet wird

`inpck`

schaltet die Paritätsprüfung in der seriellen Schnittstelle ein

`ignpar`

erkannte Paritätsfehler werden bei der Terminalausgabe ignoriert

`parmrk`

erkannte Paritätsfehler werden bei der Ausgabe mit einer 255-0-Zeichen Sequenz markiert

`istrip`

löscht das 8. Bit der ankommenden Zeichen

`inlcr`

übersetzt ankommende NEWLINE (Zeilenvorschub, ^J) in CARRIAGE RETURN (Wagenrücklauf, ^M)

`igncr`

ignoriert ankommende CARRIAGE RETURN

`icrnl`

übersetzt ankommende Wagenrücklauf in Zeilenvorschub

`iucld`



alle ankommenden Großbuchstaben werden in Kleinbuchstaben umgewandelt

ixon

der Terminaltreiber reagiert auf XON/XOFF Flowcontrol

ixoff | tandem

schaltet die automatische XON/XOFF Datenflußkontrolle für die seriellen Schnittstellen ein; sobald der freie Platz im Eingabedatenpuffer des Terminals die „Niedrigwassermarke“ von 16 Zeichen unterschreitet, wird die STOP\_CHAR(tty)-Funktion ausgelöst

ixany

erlaubt jedes Zeichen als Startzeichen nach dem Einfrieren des Datenflusses beim XON/XOFF Flowcontrol (nur ^Q mit -ixany)

imaxbel

wird vom Linux-Kernel ignoriert

## Einstellungen für die Terminalausgabe:

opost

schaltet die „Nachbehandlung“ der ankommenden Zeichen im Terminaltreiber ein; die Nachbearbeitung der Ausgabe kann mit -opost komplett abgeschaltet werden

olcuc

übersetzt Kleinbuchstaben in Großbuchstaben

ocrnl

übersetzt Wagenrücklauf in Zeilenvorschub

onlcr

übersetzt Zeilenvorschub in Zeilenvorschub und Wagenrücklauf

onocr

unterdrückt Wagenrücklauf in der ersten Spalte

onlret

Zeilenvorschub erzeugt zusätzlichen Wagenrücklauf

Die Einstellungen für verschiedenste Ausgabeverzögerungen, die im stty-Kommando erlaubt werden, stammen aus der Zeit, als noch mechanische Ausgabegeräte mit trägen Druckköpfen als Terminals benutzt wurden. Die Ausgabeverzögerungen verschaffen dem Drucker genügend Zeit, die Mechanik korrekt zu positionieren, bevor das nächste Zeichen gesendet wird. Die Parameter haben bei den modernen Ausgabegeräten unter Linux keine Bedeutung mehr und werden von den Terminaltreibern im Kernel nicht benutzt. **Lokale Einstellungen:**

isig

ermöglicht Signalzeichen für Softwareinterrupts; die Zuordnung bestimmter Signale zu Eingabezeichen ist auf der nächsten Seite erklärt

icanon

schaltet die primitiven Zeileneditorfunktionen des Terminaltreibers ein, siehe nächste Seite unter Spezialzeichen

ixexten

ermöglicht Spezialzeichen außerhalb des POSIX-Standard

**echo**  
wiederholt gelesene Zeichen auf der Ausgabe

**echoe | crterase**  
zeigt Rückschritt als Rückschritt-Leerzeichen-Rückschritt an

**echok**  
gibt einen Zeilenvorschub nach einem Killzeichen (^U) aus; auch diese Funktion wird vom aktuellen Linux-Kernel nicht benutzt

**echonl**  
ein `NEWLINE` wird im kanonischen Modus reflektiert, auch wenn das Echo für andere Zeichen abgeschaltet ist

**noflsh**  
unterdrückt das Löschen des Eingabepuffers nach einer Unterbrechung

**xcase**  
wird von den Treibern im Linux-Kernel ignoriert

**tostop**  
hält Hintergrundprozesse an, die auf das Terminal schreiben wollen

**echoctl | ctlecho**  
gibt Controlzeichen als Caret-Sequenz aus (^C für Control-C)

**echoprt | prterase**  
wird vom Linux-Kernel nicht bearbeitet

**echoke | crtkill**  
wird vom Linux-Kernel ebenfalls ignoriert

**Kombinationen von Einstellungen:** Die folgenden Parameter für `stty` werden nicht direkt an den Kernel weitergegeben. Sie sind vielmehr Zusammenfassungen häufig gebrauchter Kombinationen der bisher vorgestellten Einstellungen.

**evenp | parity**  
das gleiche wie `parenb -parodd cs7` (mit einem '-' wie `-parenb cs8`)

**oddp**  
das gleiche wie `parenb parodd c7` (mit '-' wie `-parenb cs8`)

**nl**  
das gleiche wie `icrnl` (mit '-' wie `-icrnl -inlcr -igncr`)

**ek**  
setzt die Löschzeichen auf ihre voreingestellten Werte

**sane**  
setzt alle Einstellungen auf einen Standardwert (nicht unbedingt die gleichen Werte wie beim Einschalten); das gleiche wie `cread -ignbrk brkint -inlcr -igncr icrnl -ixoff -iuclc -ixany imaxbel opost-olcuc-ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0 isig icanon iexten echo echoe echok -echonl -noflsh -xcase -tostop -echoprt echoctl echoke`; außerdem werden alle Spezialzeichen auf ihre voreingestellten Werte

zurückgesetzt

cooked

ermöglicht primitive Editorfunktionen für die Standardeingabe, mit Löschen einzelner Zeichen, Wörter oder ganzer Zeilen etc.; die Eingabe wird erst nach einem Zeilenende dem bearbeitenden Programm übergeben; das gleiche wie `brkint ignpar istrip icrnl ixon opost isig icanon`; außerdem werden die `eof` und `eol` Zeichen auf ihre voreingestellten Werte zurückgesetzt, wenn sie die gleichen wie die `min` und `time` Zeichen sind; mit dem optionalen ``-'` das gleiche wie `raw`

raw

setzt die Terminalparameter auf „rohe“ Eingabe, jedes Zeichen wird sofort und roh an das bearbeitende Programm weitergegeben; das gleiche wie `-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr -icrnl -ixon -ixoff -iuclo -ixany -imaxbel -opost -isig -icanon -xcase min 1 time 0`; mit dem optionalen ``-'` das gleiche wie `cooked`

cbreak

das gleiche wie `-icanon`

pass8

das gleiche wie `-parenb -istrip cs8`; mit ``-'` das gleiche wie `parenb istrip cs7`

litout

das gleiche wie `-parenb -istrip -opost cs8`; mit ``-'` das gleiche wie `parenb istrip opost cs7`

decctlq

das gleiche wie `-ixany`

tabs

das gleiche wie `tab0`; mit dem optionalen ``-'` das gleiche wie `tab3`

lcase | LCASE


das gleiche wie `xcase iuclo olcuo`

crt

das gleiche wie `echoe echoctl echoke`

dec

das gleiche wie `echoe echoctl echoke -ixany`; außerdem wird das Spezialzeichen ``intr'` mit `^C`, ``erase'` mit `DEL` und ``kill'` mit `^U` belegt

**Spezialzeichen:** Die Spezialzeichen können mit der Syntax ``Name = Wert'` definiert werden. Der Wert kann entweder als Tastenkombination, als hexadezimale Zahl mit ``0x'` am Anfang, als oktale Zahl mit ``0'` am Anfang oder als einfache dezimale Zahl angegeben werden. Der Wert ``^-`  oder ``undef'` schaltet ein Spezialzeichen ab. Folgende Spezialzeichen werden unterstützt:

intr

sendet ein SIGINT (`^C`)

quit

sendet ein SIGQUIT (`^\\`)

erase

	löscht das zuletzt eingegebene Zeichen (^?)
kill	
	löscht die aktuelle Zeile (^U)
eof	
	sendet ein Dateiendezeichen (beendet die Eingabe) (^D)
eol	
	Zeilenende (undef)
eol2	
	alternatives Zeilenende (undef)
start	
	fährt mit einer angehaltenen Ausgabe fort (^Q)
stop	
	hält die Ausgabe an (^S)
susp	
	sendet ein SIGSTOP an ein Terminal (^Z)
rprnt	
	erneuert den Bildschirm (^R)
werase	
	löscht das letzte Wort (^W)

lnext  
fügt das nächste Zeichen uninterpretiert ein, auch wenn es ein Spezialzeichen ist (^V)

Unabhängig vom `stty` Programm bietet der Terminaltreiber von Linux einen zweiten Zeichensatz an, der unter anderem grafische Zeichen zur Umrandung von Menüs oder Boxen enthält. Zwischen dem normalen und dem Sonderzeichensatz wird mit ^N und ^O umgeschaltet. Diese Sonderzeichen müssen selbst durch das Sonderzeichen ^V eingeleitet werden, damit sie unverändert von der Shell an das Terminal gegeben werden.

## Spezielle Einstellungen:

min *N*  
bestimmt die minimale Zeichenzahl zum Abbrechen eines Lesezyklus im cbreak Modus

time *N*  
bestimmt die Zeit in Zehntelsekunden, nach der ein Lesezyklus im cbreak Modus automatisch beendet wird, auch wenn die bei ``min'` angegebene Anzahl Zeichen noch nicht gelesen ist

rows *N*  
zeigt dem Kernel an, daß das Terminal *N* Zeilen auf dem Bildschirm darstellen kann

cols *N*  
zeigt dem Kernel an, daß das Terminal *N* Spalten darstellen kann

size  
ist keine Einstellung, sondern gibt die aktuell eingestellte Bildschirmgröße (Zeilen und Spalten) aus

line *N*

setzt die Leitungsparameter auf *N*

speed

zeigt die Leitungsgeschwindigkeit an

*N*

setzt die Eingabe- und die Ausgabegeschwindigkeit auf *Geschwindigkeit* (in Bit pro Sekunde); *Geschwindigkeit* kann dabei einer der folgenden Werte sein

0 50 75 110 134 134.5 150 200 300 600 1200 1800 2400 4800 9600 19200 38400; mit dem Wert 0 kann eine Leitung unterbrochen werden, für die `local` gesetzt ist; um höhere Übertragungsraten als 38400 bps zu erzielen, muß die serielle Schnittstelle mit dem `setserial`-Kommando umgestellt werden

ispeed *N*

setzt die Eingabegeschwindigkeit auf *N* Zeichen pro Sekunde

ospeed *N*

setzt die Ausgabegeschwindigkeit auf *N* Zeichen pro Sekunde

## Optionen:

-a

zeigt alle Einstellungen in lesbarer Form


-g

gibt alle Einstellungen in einer Form, die beim Zurücklesen (als Kommandozeilenargument) die gleichen Einstellungen reproduziert

## Beispiel:

Das folgende Kommando gibt alle Einstellungen der seriellen Schnittstelle `/dev/ttyS1` aus.

```
$ stty -a < /dev/ttyS1
speed 9600 baud; rows 0; columns 0; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D;
eol = <undef>; eol2 = <undef>; start = ^Q; stop = ^S; susp = ^Z;
rprnt = ^R; werase = ^W; lnext = ^V; min = 1; time = 0;
-parenb -parodd cs8 hupcl -cstopb cread clocal -crtscts
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr -icrnl
-ixon -ixoff -iuclc -ixany -imaxbel
opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab3
bs0 vt0 ff0 -isig -icanon -iexten -echo -echoe -echok -echonl
-noflsh -xcase -tostop -echoprt -echoctl -echoke
$ _
```

Ein weiteres Beispiel zeigt, wie das aktuelle Terminal, das zuerst durch eine offenbar fehlerhafte Eingabe  durcheinandergebracht worden ist, wieder „saniert“ werden kann:

```
$ monster -d terminal
```

```
$ pwd
/usr/local/lib
$ cd
$ stty sane
$ pwd
/home/she
$ _
```

**Autor:**

David MacKenzie

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [su](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [strace](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- 

# su

## Funktion:

**su** (superuser) ändert User- und Gruppen-ID

## Syntax:

```
su [-flmp] [-c Befehl] [-s Shell] [-login] [-fast]  
[-preserve-environment] [-command=Befehl] [-shell=Shell] [-] [ Name  
[Argument ...]]
```

## Beschreibung:

**su** startet eine neue Shell unter einer neuen Benutzerkennung (UID) und Gruppenkennung (GID). Wie bei einem neuen Login wird das Paßwort des Benutzers abgefragt. Wenn kein Name angegeben ist, wird zur ID 0 gewechselt, man ändert die ID also in die der „Superuserin“. Allein sie kann mit dem **su**-Kommando die Identität jedes beliebigen Benutzers annehmen, ohne nach einem Paßwort gefragt zu werden. Damit (und nur damit) ist es möglich, auch unter den Verwaltungsnamen wie *bin*, *news* oder *daemon* zu arbeiten, die normalerweise durch ein Sperrpaßwort gesichert werden.

Ohne weitere Optionen wechselt das **su**-Kommando das Arbeitsverzeichnis nicht, setzt aber die Umgebungsvariablen HOME und SHELL auf die neuen Werte aus /etc/passwd. Wenn die neue Identität nicht die der Superuserin ist, wird auch die LOGNAME Umgebungsvariable entsprechend verändert.

In keinem Fall wird von **su** ein Eintrag in der Datenbank /etc/utmp angelegt. Das bedeutet, daß ein „Login“ mit **su** nicht mit **last** oder **who** angezeigt wird. Insbesondere geben auch das **loginname**-Kommando und die **getlogin(3)** Bibliotheksfunktion weiterhin den ursprünglichen Loginnamen aus.

Wenn außer dem Namen weitere Argumente in der Kommandozeile angegeben sind, werden sie der Shell übergeben.

## Optionen:

-f

(fast) startet die Shell mit der Option `-f`, die bei der (t)csh das Lesen der Initialisierungsdatei unterdrückt; bei der bash wird die Pfadnamenerweiterung unterdrückt, was nicht unbedingt das gewünschte Verhalten ist

-l

(login) die Shell wird aufgerufen wie bei einem `login` als *Name*; es werden alle Umgebungsvariablen außer TERM, HOME und SHELL entfernt, der Pfad auf einen eincompilierten Wert gesetzt und ein `-` als erstes Zeichen in die Kommandozeile beim Aufruf der Shell geschrieben; daraufhin startet die Shell als Loginshell mit den entsprechenden Initialisierungen; die LOGNAME Variable wird auch für die Superuserin neu gesetzt

-p und -m

(preserve environment) erhält die alte Systemumgebung, das heißt die Umgebungsvariablen HOME, SHELL, USER und LOGNAME werden nicht verändert; es wird die in der Umgebungsvariablen SHELL bestimmte Shell gestartet, wenn nicht mit der `-s` Option eine andere Shell angegeben ist

-c *Befehl* [*Argument* ]

führt nur den *Befehl* mit dem *Argument* aus

-s *Shell*

startet die *Shell* anstelle der in der Paßwortdatei festgelegten Shell (bzw. /bin/sh)

## Siehe auch:

[newgrp](#)

## Autor:

David MacKenzie

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [sum](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [stty](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)



**Next:** [superformat](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [su](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# sum

## Funktion:

**sum** berechnet eine Prüfsumme zu einer Datei

## Syntax:

```
sum [-rs] [-sysv ] [Datei ...]
```

## Beschreibung:

**sum** liest eine Datei oder die Standardeingabe, wenn keine Datei bzw. anstelle einer Datei '-' angegeben wurde, und errechnet daraus eine 16 Bit Prüfsumme und die Dateilänge (gerundet in Kilobytes). Wenn diese Prüfsumme festgehalten wird, kann später durch den Vergleich dieser Prüfsumme mit einer neu errechneten eine eventuelle Veränderung dieser Datei festgestellt werden. Wenn die Prüfsummen gleich sind, bedeutet das aber umgekehrt nicht, daß die Dateien mit Sicherheit gleich sind!

Es gibt verschiedene Verfahren zur Prüfsummenberechnung. Das **sum**-Kommando benutzt normalerweise das bei BSD Unix übliche Verfahren. Es bietet aber auch die Möglichkeit, einen System-V-kompatiblen Algorithmus zur Prüfsummenberechnung zu verwenden.

Um eine Prüfsumme nach dem neuen, im POSIX-2 Standard festgelegten CRC Verfahren zu berechnen, gibt es das **cksum**-Kommando.

## Optionen:

**-r**

erzwingt die Berechnung nach dem bei BSD Unix üblichen Verfahren (Voreinstellung)

**-s**

die Prüfsumme wird nach dem für System 5 üblichen Verfahren berechnet; außerdem wird die Dateilänge in Blöcken zu 512 Bytes berechnet und der Dateiname mit ausgegeben

**Siehe auch:**

[cksum](#)

**Autor:**

Kayvan Aghaiepour und David MacKenzie

---

[Next](#) [Up](#) [Previous](#) [Contents](#) [Index](#)

**Next:** [superformat](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [su](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

**Next:** [sync](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [sum](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- 

# superformat

## Funktion:

**superformat** formatiert eine Diskette (low-level) in praktisch jedem denkbaren Format

## Syntax:

```
superformat [-d Gerätedatei] [-D Laufwerk] [-s Anzahl] [-H Anzahl]
[-t Anzahl] [-fm] [-dd] [-hd] [-ed] [-v Level] [-f] [-B] [-V] [-b
Nummer] [-e Nummer] [-S Sektorgröße] [-G Sektorabstand] [-F
Final-Gap] [-i Sektorsprung] [-c Chunk] [-g R/W-Gap] [-r Datenrate]
[-2] [-1] [-absolute_skew] [-sizecode Sektorgröße] [-format_gap
Sektorabstand] [-head_skew long] [-track_skew long] [-stretch
Spursprung] [-aligned_skew] [-m long] Gerätedatei
```

## Beschreibung:

**superformat** erzeugt die physikalische Datenstruktur auf Disketten. Das Programm erlaubt die Veränderung jedes denkbaren Formatparameters, es läßt sich also jedes beliebige Diskettenformat damit erzeugen. Dabei sind allerdings nicht alle Formate sinnvoll.

Der Kernel hat die Parameter für die wichtigsten 31 Formate gespeichert. Für alle anderen Formate müssen die Parameter im Floppycontroller nach jedem Diskettenwechsel ``von Hand" neu gesetzt werden. Die korrekten Parameter können unmittelbar nach dem Formatieren mit dem Programm `getfdprm` ermittelt werden. Die von `getfdprm` ausgegebenen Zahlen können in der Datei abgespeichert und jederzeit mit dem `setfdprm`-Kommando wieder in den Floppycontroller geladen werden.

`superformat` ruft nach erfolgreicher Formatierung automatisch das Programm `mformat` auf und erzeugt damit ein DOS-Dateisystem. Andere Dateisysteme können mit Programmen wie `mke2fs` oder `mkxfs` erzeugt werden.

Wenn mehrere Sektoren zu Gruppen zusammengefasst werden, also bei Einstellungen mit mehr als 21

Sektoren bei 3,5 Zoll Disketten und mehr als 18 Sektoren bei 5,25 Zoll Disketten, erzeugt superformat normalerweise Disketten im 2m-Format. Dieses Format ist ausschließlich für DOS-Dateisysteme in Zusammenarbeit mit den `mttools` geeignet. Es zeichnet sich dadurch aus, daß die erste Spur mit nur 18 (15) Sektoren formatiert wird und deshalb vom Kernel sehr schnell automatisch erkannt werden kann. Die geeigneten Parameter für die restliche Diskette werden anhand der Formatinformation im Diskettendateisystem vom `mttools`-Kommando gesetzt.

Wenn die Diskette ein anderes als das DOS-Dateisystem enthalten soll oder wenn sie als rohes `tar`-Archiv verwendet wird muß die 2m-Formtierung unterdrückt werden.

## Optionen:

`-d` *Gerätedatei*

bestimmt eine andere Gerätedatei als `/dev/fd0` zum Formatieren

`-s` *Anzahl*

legt die *Anzahl* der Sektoren pro Spur fest

`-t` *Anzahl*

legt die *Anzahl* der Spuren fest

`-H` *Anzahl*

legt die *Anzahl* der Köpfe (Diskettenseiten) fest

`-l`

unterdrückt die 2m-Formatierung

`-2`

schaltet die 2m-Formatierung ein

`-f`

unterdrückt die Verifizierung der formatierten Spuren

`-dd`

veranlaßt die Formatierung einer DD Diskette

`-hd`

veranlaßt die Formatierung einer HD Diskette

`-ed`

veranlaßt die Formatierung einer ED Diskette

## Siehe auch:

[fdformat](#) und [/dev/fd?](#)

## Autor:

Alain Knaff

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [sync](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [sum](#)



**Next:** [tac](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [superformat](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
- 

# sync

## Funktion:

**sync** schreibt die gepufferten Blöcke auf die Platte

## Syntax:

**sync**

## Beschreibung:

Linux unterhält zur Optimierung der Festplatten und Diskettenzugriffe ein Blockdepot (Cache), in dem einige Datenblöcke des Massenspeichers im Arbeitsspeicher bereitgehalten werden. Veränderungen an diesen Daten werden zuerst nur im Arbeitsspeicher vorgenommen. Mit dem Systemprogramm **sync** können die veränderten Daten sofort auf den Massenspeicher gesichert werden.

Normalerweise werden die veränderten Datenblöcke des Cache in regelmäßigen Abständen automatisch vom **update**-Dämon auf die Festplatte zurückgeschrieben.

Gelegentlich, zum Beispiel vor dem Ausschalten des Rechners, ist es sinnvoll, eine manuelle Synchronisation des Dateisystems vorzunehmen. Wird der Rechner z. B. infolge eines Defektes ohne Datensynchronisation abgeschaltet, kommt es meist zu Datenverlusten und Fehlern im Dateisystem.

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- 

# tac

## Funktion:

**tac** wie cat, nur umgekehrt

## Syntax:

```
tac [-br] [-s Trenner] [-before] [-regex] [-separator=Trenner] [Datei ...]
```

## Beschreibung:

**tac** arbeitet in vieler Hinsicht wie *cat*. Es werden die einzelnen Felder der *Datei* zeilenweise ausgegeben, allerdings das letzte zuerst. Als Feldtrenner dient das Zeilenende, wenn kein anderer angegeben ist. Der Feldtrenner wird zu dem Feld gezählt, das er abschließt, also an dessen Ende ausgegeben.

## Optionen:

**-b**

(before) der Feldtrenner wird zum darauffolgenden Feld gezählt, also an dessen Anfang ausgegeben.

**-r**

der Feldtrenner ist ein regulärer Ausdruck

**-s *Trenner***

benutzt *Trenner* als Feldtrenner

## Siehe auch:

[cat](#)

### Autor:

Jay Lepreau, David MacKenzie

---

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)



Next: [tar](#) Up: [Von GNU's, Muscheln und](#) Previous: [tac](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- 

# tail

## Funktion:

**tail** zeigt das Ende einer Datei

## Syntax:

```
tail [-c [+]N[bkm]] [-n [+]N] [-fqv] [-bytes=[+]N[bkm]] [-lines=[+]N]
[-follow] [-quiet] [-silent] [-verbose] [Datei ...]
```

```
tail [{-,+}Nbckflmqv] [Datei ...]
```

## Beschreibung:

**tail** druckt die letzten (10) Zeilen einer *Datei* oder von der Standardeingabe, wenn keine Datei angegeben wird. Ein einzelnes '-' anstelle eines Dateinamens meint ebenfalls die Standardeingabe. Werden mehrere Dateien angegeben, so wird das Ende jeder Datei mit dem Dateinamen eingeschlossen in '==>' und '<==' eingeleitet.

## Optionen:

**-c** *N*

zeigt *N* Bytes vom Ende der *Datei*; der Anzahl kann eine Einheit folgen; möglich sind:

**b**

Blöcke mit 512 Bytes

**k**

Blöcke mit Kilobytes

**m**

Blöcke mit Megabytes

-f

(follow) gibt immer wieder das Dateiende aus, dadurch kann die Entwicklung einer wachsenden Datei beobachtet werden; diese Option funktioniert nur, wenn nur eine einzige Datei angegeben ist

-n *Anzahl*

gibt *N* Zeilen aus

-q

unterdrückt die Dateinamen zu Beginn der Ausgabe

-v

druckt immer die Dateinamen zu Beginn der Ausgabe

-*Anzahl*

gibt die angegebene *Anzahl* Zeilen aus

## Siehe auch:

[head](#)

### Autor:

Paul Rubin, David MacKenzie

---

Next	Up	Previous	Contents	Index
------	----	----------	----------	-------

**Next:** [tar](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [tac](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

**Next:** [tee](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [tail](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Beispiel:](#)
  - [Siehe auch:](#)
- 

# tar

## Funktion:

**tar** (tape archiver) verwaltet Dateiarhive

## Syntax:

```
tar [-AcdrtuX] [-delete] [-b N] [-BgGhiklmMoOpPPsSvwWz] [-C  
Verzeichnis] [-f Datei] [-F Datei] [-K Datei] [-L Länge] [-N Datum]  
[-T Datei] [-V Name] [-X Datei] [0-7] [{lmh}]
```

## Beschreibung:

**tar** ist ursprünglich ein Tool zur Verwaltung von Bandarchiven. Das GNU-**tar** kann aber auch auf „rohen“ Disketten oder in normalen Dateien Archive im **tar** Format anlegen und verwalten. Normalerweise werden Archive mit **tar** nicht komprimiert. Das GNU-**tar** kann aber die Ein- und Ausgabe durch einen Kompressor leiten. Die neuen Versionen (ab 1.11.2) unterstützen sowohl **compress** als auch **gzip**.

Wenn auf der Kommandozeile keine Datei und kein Gerät angegeben ist, versucht **tar** auf die Gerätedatei eines Magnetbandgerätes zuzugreifen. Je nach Konfiguration ist das meist **/dev/tape** oder **/dev/rmt0**. Sie können eine andere Voreinstellung wählen, indem Sie die Umgebungsvariable **TAPE** mit dem Pfadnamen der entsprechenden Gerätedatei belegen.

## Optionen:

Die Optionen können aus Gründen der Kompatibilität mit anderen, älteren Versionen von **tar** auch in Abweichung von den [POSIX-Regeln](#) ohne das für Optionen übliche Minuszeichen angegeben werden. **tar** interpretiert den ersten Kommandoparameter immer als Optionsblock. Optionsargumente müssen dann im Anschluß an den Optionsblock in der Reihenfolge angegeben

werden, in der die Optionen im Optionsblock erscheinen.

Einige wenige spezielle Optionen des GNU-`tar` sind nicht als einfache Buchstabenoptionen erreichbar. Diese Schalter und Regler können Sie nur in der für GNU-Kommandos spezifischen verbalen Form erreichen.

-A

hängt ein komplettes Archiv an ein anderes Archiv an (nicht für Magnetbänder)

-C

erzeugt ein neues Archiv

-d

vergleicht das Archiv mit dem Dateisystem

- -delete ***Datei***

löscht die *Datei* aus dem Archiv (nicht für Magnetbänder)

-r

hängt Dateien an das Archiv an (nicht für Magnetbänder)

-t

zeigt den Inhalt des Archivs

-u

ersetzt Dateien, die neuer als eine bereits archivierte Version sind; ist eine Datei noch nicht archiviert, so wird sie eingefügt (nicht für Magnetbänder)

-x

kopiert *Datei* oder alle Dateien aus dem Archiv

Weitere Optionen

-atime-preserve

veranlaßt `tar`, die Zugriffszeit nach der Archivierung zurück zu setzen

-b *N*

setzt die Blockgröße auf *N*x512 Bytes (Voreinstellung ist *N*=20) 

-B

unterdrückt den Abbruch von `tar` beim Lesen unvollständiger Blöcke; zum Lesen von 4.2BSD Pipes

-C ***Verzeichnis***

wechselt während der Ausführung in das *Verzeichnis*, um von dort weitere Dateien zu archivieren

-f ***Datei***

benutzt *Datei* oder das damit verbundenen Gerät als Archiv

-F ***Datei***

bei mehrteiligen Archiven (Option -M) wird das Shellsript *Datei* ausgeführt, wenn das Medium voll ist

-G

erzeugt am Anfang des Bandarchives einen speziellen Eintrag für jedes archivierte Verzeichnis; spezielles GNU Format

-g ***Datei***

erzeugt eine *Datei* mit einer Liste der archivierten Verzeichnisse als Zeitmarke der Archivierung; wenn die *Datei* bereits existiert, werden nur die Dateien archiviert, die nach dieser Zeitmarke erzeugt oder verändert wurden (spezielles GNU Format: 1. Zeile = Zeitmarke, 1. Feld = Nr. der Partition, 2. Feld = Inode des Verzeichnisses, 3. Feld = Name des Verzeichnisses)

-h

archiviert die referenzierten Dateien anstelle der Links

-i

ignoriert Blöcke mit Nullbytes im Archiv

-k

existierende Dateien werden beim Auspacken von Archiven nicht überschrieben

-K ***Datei***

beginnt eine Aktion bei *Datei* im Archiv

-l

verhindert Archivierung von Dateien aus anderen Dateisystemen

-L ***Länge***

wartet auf Medienwechsel nach *Länge* Bytes

-m

das Datum der letzten Änderung wird nicht mitarchiviert

-M

das Archiv ist auf mehrere Medien verteilt (Multi-Volume)

-N ***Datum***

archiviert nur Dateien, die neuer sind als *Datum*

-O

benutzt das alte V7 tar-Format anstelle des ANSI Formates

-O

schreibt die Dateien in die Standardausgabe

-p

erhält die Zugriffsrechte der Dateien

-P

archiviert mit absoluten Dateinamen

-R

gibt zu jeder Meldung die Blocknummer des Archivblocks aus, von dem die Meldung verursacht wurde

-s

zeigt an, daß die Liste von Dateien im Argument die gleiche Reihenfolge hat wie die Dateien im Archiv

-T ***Datei***

holt die Namen der zu archivierenden Dateien aus *Datei*

-v

meldet jede Aktion

-v *Name*

erzeugt ein Archiv mit dem (internen) Label *Name*

-w

erwartet interaktiv Bestätigung jeder Aktion

-W

verifiziert die geschriebenen Daten im Archiv

-X *Datei*

liest aus der *Datei* Namen oder reguläre Ausdrücke von bzw. für Dateien, die nicht archiviert werden sollen


-z

erzeugt ein mit `gzip` komprimiertes Archiv

-Z

erzeugt ein mit `compress` komprimiertes Archiv

-{0..7}{lmh}

spezifiziert das Gerät und die Dichte des Speichermediums (für Diskettenarchive ohne Bedeutung); 0 ist der erste Streamer, 1 der zweite und so weiter; die Dichte bestimmt den Bandtyp 

## Beispiel:

Beispiele für die Verwendung von `tar` finden Sie [hier](#) und [hier](#).

## Siehe auch:

[cpio](#), [mt](#), `shar(1)`

## Autor:

John Gilmore

---

Next	Up	Previous	Contents	Index
------	----	----------	----------	-------

**Next:** [tee](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [tail](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

**Next:** [touch](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [tar](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# tee

## Funktion:

**tee** verzweigt die Ausgabe auf eine Datei

## Syntax:

```
tee [-ai] [-append] [-ignore-interrupts] [Datei ...]
```

## Beschreibung:

**tee** liest von der Standardeingabe und verzweigt die Ausgabe auf die Standardausgabe und *Datei*. Wird auf eine existierende Datei verzweigt, so wird sie überschrieben, anderenfalls wird sie angelegt.

Die Ausgabe von `make` bzw `gcc` beim compilieren eines Programms kann beispielsweise mit

```
`make -k 2>&1 | tee make.out'(bash-Syntax)
```

in der Datei `make.out` gesichert werden.

## Optionen:

`-a`

die *Datei* wird nicht überschrieben, sondern die Ausgabe daran angehängt

`-i`

ignoriert Interrupt Signale

## Autor:

Mike Parker, Richard M. Stallman und David MacKenzie

---

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# touch

## Funktion:

**touch** ändert die Zeitmarkierung einer Datei

## Syntax:

```
touch [-acm] [-r Referenzdatei] [-t MMDDhhmm[[CC]YY][.ss]] [-d Zeit]
[-time={atime, access, use, mtime, modify}] [-date=Zeit]
[-file=Referenzdatei] [-no-create] Datei ...
```

## Beschreibung:

**touch** setzt die Zugriffs- und die Änderungszeit der *Datei* auf die aktuelle Zeit. Wenn die *Datei* nicht existiert, wird sie erzeugt.

Wenn als erster Dateiname ein gültiges Argument für die `-t` Option übergeben wird, und keine der Optionen `-d`, `-r` oder `-t` ausdrücklich angegeben worden ist, wird dieses Argument als Zeitmarke für die folgenden Dateien verwendet.

## Optionen:

`-a`

ändert nur die Zugriffszeit

`-c`

unterdrückt die Erzeugung nicht existierender Dateien

`-d Zeit`

setzt *Zeit* anstelle der aktuellen Uhrzeit; für *Zeit* können verschiedene gebräuchliche Formate verwendet werden

`-m`

ändert nur die Änderungszeit



-r *Referenzdatei*

setzt die Zeit von *Referenzdatei* anstelle der aktuellen Zeit

-t *MMDDhhmm* [[*CC*] *YY*][.ss]

benutzt das Argument als Zeitangabe

**Autoren:** Paul Rubin, Arnold Robbins,  
Jim Kingdon, David MacKenzie und Randy Smith

---

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

**Next:** [uname](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [touch](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# tty

## Funktion:

**tty** gibt die Bezeichnung des aktuellen Terminals aus

## Syntax:

```
tty [-s] [-silent] [-quiet]
```

## Beschreibung:

**tty** gibt den Namen des Terminals inclusive Pfad aus, das die Standardeingabe repräsentiert. Der Status wird dabei wie folgt gesetzt:

- 0  
wenn die Standardeingabe ein Terminal ist
- 1  
wenn die Standardeingabe nicht ein Terminal ist
- 2  
wenn ein anderer Fehler aufgetreten ist

## Optionen:

- s  
keine Ausgabe; nur der Status wird gesetzt

## Autor:

David MacKenzie

**Next:** [uniq](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [tty](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Optionen:](#)
- 

# uname

## Funktion:

**uname** gibt Auskunft über Betriebssystem und Hardware

## Syntax:

```
uname [-snrvma] [-sysname] [-nodename] [-release] [-version]  
[-machine] [-all]
```

## Optionen:

- s  
zeigt den Betriebssystemnamen (Voreinstellung, wenn keine Option angegeben wird)
- n  
zeigt den Netzwerknamen des Systems
- r  
zeigt die Release (Versionsnummer) des Betriebssystems
- v  
zeigt das Erstellungsdatum des Betriebssystems
- m  
zeigt den Prozessortyp
- a  
zeigt alle der oben genannten Daten

## Autor:

David MacKenzie



**Next:** [wall](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [uname](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Optionen:](#)
- 

# uniq

## Funktion:

**uniq** löscht doppelte Zeilen aus einer sortierten Datei

## Syntax:

```
uniq [-cdu] [-f Nummer] [-s Nummer] [-w Nummer] [-#Nummer] [+#Nummer]
[-count] [-repeated] [-unique] [-skip-fields=Nummer]
[-skip-chars=Nummer] [-check-chars=Nummer][Eingabedatei]
[Ausgabedatei]
```

## Optionen:

-u

gibt nur die einzigartigen Zeilen aus, die keine gleichen Nachbarzeilen haben.

-d

gibt nur die Zeilen mit Doppelgängern aus

-c

gibt am Anfang jeder Zeile mit Doppelgänger die Anzahl des Vorkommens aus

-f *Nummer*

überspringt *Nummer* Felder beim Vergleich; felder sind durch Leerzeichen und TAB getrennt; führende Leerzeichen werden ignoriert

-s *Nummer*

überspringt *Nummer* Zeichen beim Vergleich; führende Leerzeichen werden ignoriert; werden Felder und Zeichen übersprungen, so werden zuerst die Felder und dann in dem erreichten Feld die Zeichen übersprungen

-w *Nummer*

es werden nur *Nummer* Zeichen verglichen; normalerweise wird die komplette Zeile (oder der Zeilenrest) verglichen

**Autor:**

Richard Stallman und David MacKenzie

---

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

**Next:** [wc](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [uniq](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
- 

# wall

## Funktion:

`wall` sendet eine Nachricht an alle aktiven Benutzer

## Syntax:

`wall`

## Beschreibung:

sendet eine Nachricht an alle aktiven Benutzer. Der Empfang von Nachrichten kann durch ``mesg no'` abgeschaltete werden. Nur die Nachrichten der Superuserin (Ruth) können nicht abgeschaltet werden.

Der Nachrichtentext wird von der Standardeingabe gelesen, und mit einem Dateiendezeichen (^D) abgeschlossen.

Das Kommando `wall` wird vor allem für Hinweise der Systemverwalterin, z. B. vor dem Abschalten des Rechners benutzt.

Next: [who](#) Up: [Von GNU's, Muscheln und](#) Previous: [wall](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# WC

## Funktion:

**wc** zählt die Anzahl von Zeichen, Wörtern oder Zeilen

## Syntax:

**wc** [-clw] [-bytes] [-chars] [-lines] [-words] [*Datei* ...]

## Beschreibung:

**wc** zählt in jeder *Datei* oder in der Standardeingabe die Zeilen, die Wörter und die Zeichen. Für jede Datei wird eine Zeile mit den Zahlen, gefolgt vom Dateinamen, ausgegeben. Wird mehr als eine Datei angegeben, wird zusätzlich die Summe der einzelnen Werte in der letzten Zeile ausgegeben.

Ohne weitere Optionen gibt wc alle drei Werte aus

## Optionen:

- l  
gibt nur die Anzahl der Zeilen aus
  - w  
gibt nur die Anzahl der Wörter aus
  - c  
gibt nur die Anzahl der Zeichen aus
-



## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# who

## Funktion:

**who** gibt Information über die aktiven Anwender des Systems

## Syntax:

**who** [-imqsuwHT] [*Datei*] [am i]

## Beschreibung:

**who** zeigt die Namen, das Terminal und die Loginzeit der aktiven Benutzer. Außerdem gibt es Antwort auf die existentielle Frage „**wer bin ich?**“!

Wird eine *Datei* angegeben, so wird die Information über die aktiven Anwender aus dieser Datei genommen. Voreingestellt ist die Datei `/etc/utmp`.

## Optionen:

- i  
zeigt die Dauer der aktuellen Sitzungen
- m  
gibt den eigenen Benutzernamen mit eMail Adresse aus
- q  
gibt nur die Benutzernamen und die Gesamtzahl der aktiven Benutzer aus
- s  
Ausgabe im Standardformat
- u  
wie -i
- w

markiert alle Benutzer mit `+', die eine Nachricht empfangen können, oder mit einem `-', wenn der Empfang von Nachrichten abgeschaltet wurde

-H

gibt zusätzlich eine Kopfzeile mit der Bedeutung der einzelnen Spalten aus

-T

wie -w

## **Autor:**

jlal, überarbeitet von David MacKenzie

---

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

**Next:** [Die Kommandos für root](#) **Up:** [Von GNU's, Muscheln und](#) **Previous:** [who](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
- 

# write

## Funktion:

**write** schreibt eine Nachricht auf das Terminal eines anderen Benutzers

## Syntax:

**write** *Name* ...

## Beschreibung:

**write** schreibt auf das Terminal von *Name*. Die Nachricht wird von der Standardeingabe gelesen. Wenn von der Tastatur gelesen wird, muß die Eingabe mit `CONTROL-D` abgeschlossen werden. Mit ``mesg no'` kann das eigene Terminal vor fremdem Beschreiben geschützt werden.

Wenn das Kommando unter dem Namen `wall` aufgerufen wird, schreibt es an alle eingeloggten Benutzer.

Die Superuserin (Ruth) kann auch an die Anwender schreiben, deren Terminal vor dem Beschreiben geschützt ist.

## Autor:

Peter Orbaek

# Die Kommandos für root

---

- [chown](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- [elvprsv](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
- [fdisk](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- [fsck \(Front-End\)](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- [fsck.ext2 \(e2fsck\)](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)

- [fsck.minix \(fsck\)](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- [fsck.xiafs \(xfck\)](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- [insmod](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- [mkboot](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Beispiel:](#)
  - [Siehe auch:](#)
- [mkfs \(Front-End\)](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- [mkfs.ext2 \(mke2fs\)](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)

- [mkfs.minix \(mkfs\)](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- [mkfs.xiafs \(mkxfs\)](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- [mknod](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- [mkswap](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Beispiel:](#)
  - [Optionen:](#)
- [mount](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- [rdev](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- [rmmod](#)
  - [Funktion:](#)
  - [Syntax:](#)

- [Beschreibung:](#)
- [Siehe auch:](#)
- [rpm](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Installation neuer Pakete](#)
  - [Upgrade existierender Software](#)
  - [Suche nach Informationen über ein Softwarepaket](#)
  - [Löschen einer installierten Software](#)
  - [Verifizieren der Integrität einer Softwareinstallation](#)
  - [Reparieren einer fehlerhaften Installation](#)
  - [Erzeugen neuer RPM-Pakete](#)
  - [Siehe auch:](#)
- [shutdown](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- [umount](#)
  - [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- 

# chown

## Funktion:

**chown** (change owner) ändert den Eigentümer der Datei(en)

## Syntax:

**chown** [-Rcfv] [-recursive] [-changes] [-silent] [-quiet] [-verbose] [*Besitzerin*][:.][*Gruppe*] *Datei* ...

## Beschreibung:

Normalerweise ist die Erzeugerin einer Datei bzw. eines Verzeichnisses auch deren/dessen Eigentümerin (die besondere Bedeutung der Eigentumsrechte ist [hier](#) erklärt). Mit **chown** kann die Systemverwalterin (*ruth*) die Eigentümerin und die Gruppe einer Datei oder eines Verzeichnisses ändern. *Besitzerin* und *Gruppe* können als Name oder Kennzahl angegeben werden. Name und Zahl müssen mit den entsprechenden Einträgen in `/etc/passwd` und `/etc/group` übereinstimmen.

Das **chown**-Kommando kann auch von Benutzerinnen ohne Privilegien ausgeführt werden, allerdings kann dann nur die Gruppe geändert werden. Weil es zu diesem speziellen Zweck das separate Kommando **chgrp** gibt, kann die Ausführung des **chown** Kommandos der Superuserin vorbehalten bleiben.

## Optionen:

-c

(changes) es werden (nur) die Dateien angezeigt, deren Besitzerin tatsächlich verändert wird

-f

(force) Fehlermeldungen wegen fehlgeschlagener Änderungsversuche werden unterdrückt

-v

(verbose) alle Aktionen werden angezeigt



-R

(recursive) die Besitzerin aller Dateien in den Unterverzeichnissen wird ebenfalls geändert

## Siehe auch:

[chgrp](#) und [chmod](#)

### Autor:

David MacKenzie

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [elvprsv](#) **Up:** [Die Kommandos für root](#) **Previous:** [Die Kommandos für root](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
- 

# elvprsv

## Funktion:

**elvprsv** - Rettet die Temporärdateien, die **elvis** in `/tmp` erstellt.

## Syntax:

**elvprsv** [*„warum elvis abstürzte“*] */tmp/Dateiname...*

**elvprsv** -R */tmp/Dateiname...*

## Beschreibung:

Der Editor **elvis** bearbeitet eine Textdatei nicht im Arbeitsspeicher sondern in einer temporären Datei im Verzeichnis `/tmp`. Wenn der Prozeß **elvis** aus irgendeinem Grund abgebrochen wird, bevor der Text regulär gesichert worden ist, bleibt diese Datei zurück. **elvprsv** sichert und archiviert solche zurückgebliebenen Temporärdateien und erlaubt so meistens die Rekonstruktion der durch die Programmunterbrechung verlorenen Arbeit.

Die von **elvprsv** gesicherten Temporärdateien befinden sich in `/var/preserve`. Die ebenfalls in diesem Verzeichnis enthaltene Indexdatei erlaubt die Zuordnung der gesicherten Temporärdateien zu den zum Zeitpunkt des Absturzes editierten Dateien.

Um die verlorenen Änderungen zu restaurieren gibt es das Benutzerkommando **elvrec**.

Bei geeigneter Installation sollte es unnötig sein, **elvprsv** von der Kommandozeile aus zu starten. **elvis** startet das Programm selbst, solange er ein terminierendes Signal abfangen kann. Damit nach einem vollständigen Systemabsturz die verbliebenen Temporärdateien automatisch gesichert werden, sollte **elvprsv** automatisch bei jedem Systemstart von einem der Initialisierungsscripte in `/etc/rc.d` aufgerufen werden.

Mit der Option **-R** kann **elvprsv** die zum Zeitpunkt des Absturzes editierte Datei direkt restaurieren ohne sie in `/var/preserve` zwischenzuspeichern.

## Autor:

[Next](#) [Up](#) [Previous](#) [Contents](#) [Index](#)

**Next:** [fdisk](#) **Up:** [Die Kommandos für root](#) **Previous:** [chown](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

**Next:** [fsck \(Front-End\)](#) **Up:** [Die Kommandos für root](#) **Previous:** [elvprsv](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- 

# fdisk

## Funktion:

**fdisk** ist der Partitionstabelleneditor für Linux

## Syntax:

**fdisk** [-lv] -s *Partition*] [*Gerätedatei*]

## Beschreibung:

**fdisk** teilt eine formatierte Festplatte in verschiedene Partitionen ein, löscht bestehende Partitionen oder gibt vorhandenen Partitionen eine andere Systemkennung.

Mit **fdisk** wird nicht eine einzelne Partition, sondern die ganze (rohe) Festplatte bearbeitet. Die Gerätedatei ist also `/dev/hda` für die erste `normale' Festplatte, `/dev/hdb` für die zweite `normale' Festplatte, `/dev/sda` für die erste SCSI-Festplatte, `/dev/sdb` für die zweite SCSI-Festplatte usw. ...

Als normale Festplatten werden von Linux sowohl AT-Bus-Platten als auch Festplatten für 16-Bit-RLL- oder MFM-Controller unterstützt. Mit entsprechender Kernelkonfiguration kann auch ein 8-Bit-XT-Controller verwendet werden.

## Optionen:

-v  
gibt die Versionsnummer aus

-l  
gibt eine Liste der Partitionstabellen für alle IDE- und SCSI-Festplatten aus

-s *Partition*  
wenn die angegebene *Partition* keine DOS-Partition ist, wird die Größe dieser Partition

ausgegeben (dieser Wert kann beispielsweise als Eingabe für das Programm `mkfs.minix` verwendet werden)

## Siehe auch:

Eine ausführliche Beschreibung des `fdisk`-Kommandos ist im Kapitel „[Die Festplatte partitionieren](#)“ zu finden.

## Autor:

A. V. Le Blanc, Rik Faith

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [fsck \(Front-End\)](#) **Up:** [Die Kommandos für root](#) **Previous:** [elvprsv](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

**Next:** [fsck.ext2 \(e2fsck\)](#) **Up:** [Die Kommandos für root](#) **Previous:** [fdisk](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- 

# fsck (Front-End)

## Funktion:

**fsck** steuert als Front-End die Prüfung und Reparatur aller Linux-Dateisysteme

## Syntax:

**fsck** [-AVRTNPs] [-t *Typ*] [*Optionen*] *Dateisystem*

## Beschreibung:

**fsck** ist das von den verschiedenen Linux-Dateisystemen unabhängige Front-End zum Prüfen und Reparieren der Dateisystem-Struktur.

Als Front-End übernimmt **fsck** die Arbeit am konkreten Dateisystem nicht selbst, sondern ruft das spezifische File-System-Check-Programm für das angegebene Dateisystem auf. Die Zuordnung eines Dateisystemtyps zu einer bestimmten Partition findet über die Datei `/etc/fstab` oder durch eine Kommandozeilenoption statt.

Als Kommandozeilenargument kann das zu prüfende Dateisystem entweder als Gerätedatei oder durch seinen Aufsetzpunkt (mount point) angegeben werden. Im Fall der Angabe durch den Aufsetzpunkt muß die Verknüpfung mit einer bestimmten Partition in der Datei `/etc/fstab` festgelegt sein.

Nach den eigenen Optionen erlaubt das **fsck**-Front-End die Übergabe weiterer Optionen, die an das spezifische File-System-Check-Programm weitergegeben werden. Die Optionen `-a`, `-l`, `-r`, `-s` und `-v` werden von den meisten dieser Programme unterstützt, Einzelheiten finden Sie bei den entsprechenden Kommandobeschreibungen.

Um sie automatisch aufrufen zu können, erwartet das **fsck**-Front-End die spezifischen File-System-Checker unter Namen der Art `fsck.Typ`, also beispielsweise `fsck.ext2` für Extended-2-Dateisysteme oder `fsck.minix` für „alte“ Minix-Dateisysteme. Der

File-System-Standard sieht das Verzeichnis `/sbin` als Installationsverzeichnis für all diese Programme vor. Das `fsck`-Front-End sucht aber auch in den Verzeichnissen `/etc/fs` und `/etc` nach den Programmdateien.

Der Name `fsck` für das Front-End leitet sich Unix-typisch aus der von Vokalen befreiten Abkürzung für *file system check* ab. Zu Beginn der Linux-Entwicklung gab es als einziges das von Minix übernommene Dateisystem, deshalb hat das Kommando zur Prüfung eines Minix-Dateisystems ursprünglich den Namen `fsck` gehabt. Mit der Einführung des Front-Ends muß das alte `fsck` seinen angestammten Namen zugunsten der einheitlichen Namenskonvention abgeben.

Um die Zusammenarbeit mit anderen Programmen zu erleichtern, gibt das `fsck`-Front-End über den Exit-Status verschiedene Informationen zum Zustand des Dateisystems und zur Programmausführung weiter:

0

Es wurde kein Fehler im Dateisystem gefunden.

1

Es wurden Fehler im Dateisystem gefunden und korrigiert.

2

Es wurden schwerwiegende Fehler im Dateisystem gefunden und korrigiert. Das System sollte rebootet werden.

4

Es wurden Fehler im Dateisystem gefunden, aber nicht korrigiert.

8

Es ist ein Fehler bei der Programmausführung aufgetreten.

16

Falsche Benutzung (Fehler in der Kommandozeile?).

128

Es ist ein Fehler in einer Funktion der Shared-Libraries aufgetreten.

Diese Fehlercodes können beispielsweise in einem `rc`-Shellscript zur Systeminitialisierung abgefragt werden, um damit den weiteren Ablauf des Programms zu bestimmen. Wenn mit der Option `-A` mehrere Dateisysteme geprüft werden, ist der Status des `fsck`-Front-Ends die logische ODER-Verknüpfung aller Rückgabewerte der einzelnen File-System-Checker.

## Optionen:

`-A`

(All) veranlaßt `fsck`, alle in der Datei `/etc/fstab` aufgeführten Linux-Dateisysteme zu prüfen; diese Option schließt die Option `-t` zur direkten Angabe eines Dateisystemtyps aus, stattdessen werden die Typen aus der `fstab` gelesen

`-R`

zusammen mit der Option `-A` wird das Rootfilesystem nicht geprüft (sinnvoll wenn es schon read-write gemountet ist)

`-T`

unterdrückt die Ausgabe des Programmnamens (Titel) beim Starten

-N

es wird nur angezeigt, was `fsck` ohne diesen Schalter machen würde

-P

zusammen mit der Option `-A` wird das Rootfilesystem parallel zu den übrigen Partitionen geprüft (unsicher!)

-V

veranlaßt `fsck`, Informationen über den Programmverlauf auszugeben; wenn diese Option mehr als einmal angegeben ist, werden die spezifischen File-System-Checker nicht wirklich ausgeführt

-S

verhindert die parallele Prüfung mehrerer Partitionen

-t *Typ*

spezifiziert den Dateisystemtyp der zu prüfenden Partition; wenn diese Option nicht angegeben ist, versucht das `fsck`-Front-End, den Typ aus der Datei `/etc/fstab` zu bestimmen

## Siehe auch:

[mkfs](#), [fsck.ext2](#), `fsck.minix(8)`, `fsck.xiafs(8)`

## Autor:

David Engel, Fred van Kempen

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [fsck.ext2 \(e2fsck\)](#) **Up:** [Die Kommandos für root](#) **Previous:** [fdisk](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)



**Next:** [fsck.minix \(fsck\)](#) **Up:** [Die Kommandos für root](#) **Previous:** [fsck \(Front-End\)](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- 

# fsck.ext2 (e2fsck)


## Funktion:

**e2fsck** überprüft und repariert ein ext2-Dateisystem

## Syntax:


**e2fsck** [-pacnyrdfvtFV] [-b *Nummer*] [-B *Größe*] [-l *Datei*] *Gerätedatei*

## Beschreibung:

**e2fsck** ist das Wartungsprogramm für Extended-2-Dateisysteme. Es prüft die Konsistenz des Dateisystems und repariert Fehler. Für die Zusammenarbeit mit dem neuen **fsck**-Front-End muß das Programm unter dem Namen `fsck.ext2` installiert sein.  In der Kommandobeschreibung steht `e2fsck` synonym für `fsck.ext2`.

Die *Gerätedatei* gibt die Partition an, auf der das Dateisystem geprüft werden soll. Wenn diese Partition kein ext2fs enthält, bricht das Kommando automatisch ab.

Wenn das valid-Flag der ext2fs-Partition gesetzt ist, wird das Dateisystem normalerweise nicht geprüft ([Siehe valid-Flag](#)).

Dateien, deren Inodes in keinem Verzeichnis eingetragen sind, werden von **e2fsck** im Verzeichnis `lost+found` mit der Inode-Nummer eingetragen und können so „gerettet“ werden. 

Um die Zusammenarbeit mit anderen Programmen zu erleichtern, gibt **e2fsck** über den Exit-Status verschiedene Informationen weiter:

0

Kein Fehler im Dateisystem gefunden.

1

Fehler im Dateisystem gefunden und korrigiert.

2

Schwerwiegende Fehler im Dateisystem gefunden und korrigiert. das System sollte rebootet werden, wenn das Dateisystem aufgesetzt war.

4

Fehler im Dateisystem gefunden und unkorrigiert gelassen.

8

Fehler bei der Programmausführung aufgetreten.

16

Falsche Benutzung (Fehler in der Kommandozeile?).

128

Fehler in den Shared-Libraries.

Diese Fehlercodes können beispielsweise in einem Shellscript abgefragt werden, um damit den weiteren Ablauf des Programms zu bestimmen.

## Optionen:

-a

(automatic) repariert alle gefundenen Fehler automatisch; diese Option ist durch `-p` ersetzt worden und daher obsolet

-b *Nummer*

liest den Superblock aus dem mit der *Nummer* bezeichneten Partitionsblock

-B *Größe*

gibt die Größe einer Blockgruppe zwischen zwei Kopien des Superblocks an; beim ext2fs werden Sicherungskopien des Superblocks normalerweise alle 8192 Blöcke angelegt

-l *Datei*

liest die Liste schlechter Blöcke (*bad blocks*) aus der angegebenen *Datei*, diese Blöcke werden automatisch als benutzt markiert

-c

(check) durchsucht das Dateisystem nach schlechten Blöcken

-d

(debug) gibt zusätzliche Information über den Status des Programms auf die Standardfehlerausgabe

-f

(force) erzwingt die Überprüfung eines Dateisystems, auch wenn das valid-Flag gesetzt ist ([Siehe valid-Flag](#))

-n

(no) veranlaßt `fsck`, das Dateisystem nur zum Lesen zu öffnen und alle Fragen eines interaktiven Laufs zu verneinen

-p

(preen) veranlaßt die automatische Reparatur aller gefunden Fehler ohne interaktive Aktion

-r

diese Option ist wirkungslos und nur zur Kompatibilität mit älteren Versionen vorhanden

-t

(time) gibt statistische Information über die Performance des Programms aus

-v

(verbose) gibt Information über den Status des Programms und über das Dateisystem aus

-V

(Version) zeigt die Versionsnummer des Programms an

-y

(yes) veranlaßt `e2fsck`, alle Fragen eines interaktiven Laufs automatisch zu bejahen.

## Siehe auch:

[mkfs.ext2](#), [fsck](#)

## Autor:

Remy Card, Linus Torvalds und Wayne Davidson

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [fsck.minix \(fsck\)](#) **Up:** [Die Kommandos für root](#) **Previous:** [fsck \(Front-End\)](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# fsck.minix (fsck)

## Funktion:

**fsck.minix** prüft die Konsistenz eines Minix-Dateisystems

## Syntax:

**fsck.minix** [-larvsm] *Geräte-datei*

## Beschreibung:

**fsck.minix** überprüft ein Minix-Dateisystem auf Blöcke, die belegt, aber keiner Datei zugeordnet sind, und auf korrekte Vernetzung der Inodes. Wenn ein Fehler festgestellt wird, kann er mit **fsck.minix** auch repariert werden.

Fehler im Dateisystem entstehen leicht durch unkontrolliertes Abschalten des Rechners. Dabei gehen die letzten Veränderungen des Dateisystems verloren, die aus dem Speicher nur bei einem *sync* auf die Festplatte geschrieben werden.

**fsck.minix** kann die Integrität des Dateisystems meistens wiederherstellen, allerdings gehen dabei die Daten der betroffenen Dateien verloren. Damit die Veränderungen am Dateisystem auch tatsächlich wirksam werden, darf **fsck.minix** mit den Optionen **-a** und **-r** nur auf abgesetzte Dateisysteme angewandt werden. Anderenfalls würden die reparierten Strukturen bei der nächsten Synchronisierung wieder durch die defekten aus dem Blockdepot im Arbeitsspeicher ersetzt werden.

Das **fsck.minix**-Programm gibt einige Informationen über den Zustand des Dateisystems als Exit-Status an das aufrufende Programm zurück:

0

Kein Fehler im Dateisystem gefunden.

3

Fehler im Dateisystem gefunden und korrigiert. Das System sollte rebootet werden, wenn das Dateisystem aufgesetzt war.

4

Fehler im Dateisystem gefunden und unkorrigiert gelassen.

8

Fehler bei der Programmausführung aufgetreten.

16

Falsche Benutzung (Fehler in der Kommandozeile?).

Mit diesen Statuswerten können beispielsweise in einem Shellscript die weiteren Programmschritte gesteuert werden.

## Optionen:

-l

(list) zeigt die Dateien und Verzeichnisse auf dem Device (Partition einer Festplatte oder Diskette)

-a

(automatic) repariert alle gefundenen Fehler automatisch

-r

(repair) repariert gefundene Fehler einzeln nach Rückfrage

-v

(verbose) gibt Information über das Dateisystem aus

-s

(superblock) zeigt den Superblock des Dateisystems

-m

(mode) gibt „Mode not cleared“-Warnungen aus, wozu auch immer

## Autor:

Linus Torvalds

---

[Next](#) [Up](#) [Previous](#) [Contents](#) [Index](#)

**Next:** [fsck.xiafs \(xfck\)](#) **Up:** [Die Kommandos für root](#) **Previous:** [fsck.ext2 \(e2fsck\)](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

**Next:** [insmod](#) **Up:** [Die Kommandos für root](#) **Previous:** [fsck.minix \(fsck\)](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- 

# fsck.xiafs (xfck)

## Funktion:

**fsck.xiafs** testet und repariert das xiafs Dateisystem

## Syntax:

**fsck.xiafs** [-akrs] *Gerätedatei*

## Beschreibung:

**fsck.xiafs** (**xfck**) checkt das xiafs. Das Programm findet Fehler im Dateisystem und kann sie reparieren. Es werden fehlerhafte Verzeichniseinträge erkannt und gelöscht, doppelt belegte Blöcke gefunden und fehlerhafte Linknummern in Inodes korrigiert.

Um mit dem **fsck**-Front-End zusammenarbeiten zu können, muß das ursprünglich unter dem Namen **xfck** verbreitete Kommando (auch) unter dem Namen **fsck.xiafs** installiert sein. Das Verhalten des Programms ändert sich durch die Umbenennung nicht.

## Optionen:

-a

(automatic) repariert die gefundenen Fehler automatisch

-k

(kernel image) liest das Kernelimage aus dem reservierten Systembereich und schreibt es in die Standardausgabe

-r

(repair) repariert das Dateisystem interaktiv; jede Aktion muß einzeln bestätigt werden

-s

(super block) prüft den Superblock und zeigt dessen Inhalt an

## Siehe auch:

[fsck](#) und [e2fsck](#)

## Autor:

Frank Q. Xia

---

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- 

# insmod

## Funktion:

**insmod** fügt zur Laufzeit Kernelmodule ein

## Syntax:


```
insmod [-fkmsvx] [-o Name] Objektdatei [Symbol=Wert[,Wert]\...]
```

## Beschreibung:

**insmod** fügt in den Linux-Kernel zur Laufzeit Module ein. Solche Laufzeitmodule erweitern den Kernel um spezielle Gerätetreiber oder andere Funktionsgruppen, die nur vorübergehend benötigt werden oder zu Testzwecken schnell auswechselbar sein sollen. Die mit **insmod** geladenen Module lassen sich mit **rmmod** wieder aus dem Kernel entfernen.

Wenn der Objektname ohne Pfadangabe und ohne den Suffix `.o` angegeben wird, sucht **insmod** automatisch in den Modulverzeichnissen des laufenden Kernels. Außerdem kann der Suchpfad in der Umgebungsvariablen `MODPATH` erweitert werden.

Zum Laden von Modulstacks steht das Kommando **modprobe** zur Verfügung. Automatisches Handling der Kernelmodule wird durch den Kernealdämon **kerneld** angeboten.

Mit **insmod** können alle exportierten statischen Variablen (Symbole) verändert werden. Damit lassen sich Module initialisieren, wie die fest eingebundenen Gerätetreiber durch die Bootparameter. Sie finden eine genaue Beschreibung der Modulinitialisierung ab Seite  des Handbuches.

## Optionen:

**-f**

(force) erzwingt das Laden von Modulen, die für andere Kernelversionen erzeugt worden sind und/oder die Versionsinformationen der Symbole nicht zueinander passen



-k

(kernel) wird von modprobe benutzt, wenn das Modul im Auftrag des Kerneldämons geladen wird

-m

(map) veranlaßt insmod, beim Laden des Moduls allerlei Information zum Debuggen des Moduls in den Standardfehlerkanal zu schreiben

-s

(syslogd) veranlaßt insmod, Debuginformation durch den syslogd protokollieren zu lassen

-v

(verbose)

-x

(no-export) verhindert den Export der externen Symbole aus dem Modul


-o *Name*

veranlaßt insmod, das Modul unter dem angegebenen *Namen* anstelle der aus dem Dateinamen abgeleiteten Bezeichnung einzubinden

***Symbol=Wert***

belegt das *Symbol* mit einem neuen *Wert*

## Siehe auch:

depmod(8), kernel(8), ksyms(1), lsmod(1), modprobe(8), rmmod(8) und das Kapitel über Module ab Seite 

## Autor:

Bas Laarhoven, Jon Tombs und Bjorn Ekwall

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [mkboot](#) **Up:** [Die Kommandos für root](#) **Previous:** [fsck.xiafs \(xfck\)](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

**Next:** [mkfs \(Front-End\)](#) **Up:** [Die Kommandos für root](#) **Previous:** [insmod](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Beispiel:](#)
  - [Siehe auch:](#)
- 

# mkboot

## Funktion:

**mkboot** schreibt ein Kernelimage in den reservierten Systembereich einer xiafs-Partition und macht sie bootfähig

## Syntax:

**mkboot** [-fr *Rootpartition*] *Bootpartition*

**mkboot** -M *Festplatte* [-Nummer *Betriebssystemname*]

## Beschreibung:

**mkboot** installiert ein Kernelimage in dem reservierten Systembereich einer xiafs Partition und macht sie bootfähig. Das Kernelimage wird von der Standardeingabe gelesen.

Das xiafs kann den Bereich am Anfang einer Partition als Systembereich reservieren. Mit dem **mkboot**-Kommando kann ein Kernelimage in diesen Bereich geschrieben werden. Außerdem wird von **mkboot** ein Bootloader in den Bootsektor dieser Partition geschrieben. Damit ist diese Partition bootfähig. Um sie aktivieren zu können, muß es sich um eine primäre Partition der ersten Festplatte handeln.

Die Bootpartition muß nicht mit der Rootpartition übereinstimmen. Es ist sogar möglich, eine minimal kleine Partition als (primäre) Bootpartition einzurichten, ohne daß ein Dateisystem darauf eingerichtet wird.

Das **mkboot**-Programm kann auch den Masterbootrecord (MBR) der Festplatte verändern. Der MBR enthält neben der Partitionstabelle für die vier primären Partitionen einen Loader, der vom BIOS automatisch bei jedem Systemstart aufgerufen wird (wenn nicht von Diskette gebootet wird). Der vom MS-DOS bekannte MBR sucht in der Partitionstabelle die aktive Partition und startet den

Bootloader, der seinerseits das Betriebssystem lädt. Mit `mkboot` kann anstelle dieses MBR ein Bootselector installiert werden, der sich beim Systemstart mit einem kleinen Menü meldet und die Auswahl zwischen mehreren Bootpartitionen ermöglicht.

## Optionen:

### **-f *Rootpartition Bootpartition***

installiert den Kernel auf der Bootpartition (mit `xiafs`); der Kernel benutzt die angegebene *Rootpartition* als Rootfilesystem

### **-r *Rootpartition Bootpartition***

schreibt den Kernel auf die rohe *Bootpartition* (ohne `xiafs`); der Kernel benutzt die *Rootpartition* als Rootfilesystem

### **-M *Festplatte* [ *-Partitionsnummer Name...*]**

schreibt einen neuen Masterbootrecord (MBR) auf die *Festplatte*; die *Partitionsnummer* bezeichnet die (primäre) Partition, der *Name* ist der Eintrag im Bootmenü

Das `mkboot`-Kommando kann auch den normalen DOS MBR schreiben. Dazu muß mit der Option `-M` nur die Festplatte ohne Partitionsnummer und Name angegeben werden.

## Beispiel:

Mit den Kommandos

```
# mkxfs -k 512 /dev/hda2 81920
# mkboot /dev/hda2 < zImage
# _
```

wird auf der 80 Megabyte großen zweiten Partition der ersten Festplatte (`/dev/hda2`) ein `xiafs`-Dateisystem eingerichtet und ein Systembereich von 512 Kilobytes reserviert. Danach wird die Kerneldatei `zImage` in diesem Systembereich installiert und die Partition bootfähig gemacht (aber nicht aktiviert).

Mit dem Kommando

```
# mkboot -M /dev/hda -2 LINUX -1 MS-DOS
# _
```

wird der Masterbootrecord der ersten Festplatte ersetzt. Der Bootselector von Frank Q. Xia kann danach sowohl die zweite Partition mit Linux als auch die erste Partition mit MS-DOS booten. Wenn der Rechner das nächste Mal von Festplatte bootet, erscheint ein Menü mit diesen Einträgen. Wenn nach 15 Sekunden keine Auswahl getroffen wurde, benutzt der Loader automatisch den ersten Eintrag, in diesem Fall also LINUX auf der zweiten Partition.

## Siehe auch:

Die Installation von [LILO](#).

**Autor:**

[Next](#) [Up](#) [Previous](#) [Contents](#) [Index](#)

**Next:** [mkfs \(Front-End\)](#) **Up:** [Die Kommandos für root](#) **Previous:** [insmod](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

**Next:** [mkfs.ext2 \(mke2fs\)](#) **Up:** [Die Kommandos für root](#) **Previous:** [mkboot](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- 

# mkfs (Front-End)

## Funktion:

**mkfs** steuert als Front-End die Einrichtung von Linux-Dateisystemen

## Syntax:

**mkfs** [-V] [-t *Typ*] [*Optionen*] *Dateisystem*

## Beschreibung:

**mkfs** ist das von dem konkreten Dateisystemtyp unabhängige Front-End zur Erzeugung eines Dateisystems auf einer Festplattenpartition oder auf einer formatierten Diskette.

Als Front-End übernimmt **mkfs** die tatsächliche Erzeugung des Dateisystems nicht selbst, sondern ruft das spezifische Programm zur Einrichtung des geforderten Dateisystems auf. Die Zuordnung eines Dateisystemtyps zu einer Festplattenpartition findet über eine Kommandozeilenooption oder durch die Datei `/etc/fstab` statt.

Als Kommandozeilenargument kann das zu prüfende Dateisystem entweder als Gerätedatei oder durch seinen Aufsetzpunkt (mount point) angegeben werden. Im Fall der Angabe durch den Aufsetzpunkt muß die Verknüpfung mit einer bestimmten Partition in der Datei `/etc/fstab` festgelegt sein.

Nach den eigenen Optionen erlaubt das **mkfs**-Front-End noch die Angabe weiter Optionen, die an das spezifische Programm zur Erzeugung des Dateisystems weitergegeben werden. Die Optionen `-c`, `-l` und `-v` werden von den meisten dieser Programme unterstützt, Einzelheiten finden Sie bei den entsprechenden Kommandobeschreibungen.

Um sie automatisch aufrufen zu können, erwartet das **mkfs**-Front-End die spezifischen Programme zur Erzeugung der Dateisysteme unter Namen der Art `mkfs.Typ`, also beispielsweise `mkfs.ext2` für Extended-2-Dateisysteme oder `mkfs.minix` für „alte“ Minix-Dateisysteme. Der

File-System-Standard sieht das Verzeichnis `/sbin` als Installationsverzeichnis für all diese Programme vor. Das `fsck`-Front-End sucht aber auch in den Verzeichnissen `/etc/fs` und `/etc` nach den Programmdateien.

Der Name `mkfs` für das Front-End leitet sich Unix-typisch aus der von Vokalen befreiten Abkürzung für *make file system* ab. Zu Beginn der Linux-Entwicklung gab es als einziges das von Minix übernommene Dateisystem, deshalb hat das Kommando zur Erzeugung eines Minix-Dateisystems ursprünglich den Namen `mkfs` gehabt. Mit der Einführung des Front-Ends muß das alte `mkfs` seinen angestammten Namen zugunsten der einheitlichen Namenskonvention abgeben.

Um die Zusammenarbeit mit anderen Programmen zu erleichtern, gibt das `mkfs`-Front-End über den Exit-Status verschiedene Informationen zum Zustand des Dateisystems und zur Programmausführung weiter:

0

Die Erzeugung des Dateisystems konnte ohne Fehler durchgeführt werden.

8

Es ist ein Fehler bei der Programmausführung aufgetreten.

16

Es ist ein Fehler bei der Benutzung aufgetreten (Fehler in der Kommandozeile?).

128

Es ist ein Fehler in einer Funktion der Shared-Libraries aufgetreten.

## Optionen:

**-V**

veranlaßt `mkfs`, Informationen über den Programmverlauf auszugeben; wenn diese Option mehr als einmal angegeben ist, wird das spezifische Programm zur Erzeugung eines Dateisystems nicht wirklich ausgeführt

**-t *Typ***

spezifiziert den Dateisystemtyp des zu erzeugenden Dateisystems; wenn diese Option nicht angegeben ist, versucht das `fsck`-Front-End, den Typ aus der Datei `/etc/fstab` zu bestimmen; wenn kein spezifisches Dateisystem bestimmt werden kann, wird ein Minix-Dateisystem erzeugt

## Siehe auch:

[mkfs.ext2](#), `mkfs.minix(8)`, `mkfs.xiafs(8)`

## Autor:

David Engel, Fred van Kempen

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [mkfs.ext2 \(mke2fs\)](#) **Up:** [Die Kommandos für root](#) **Previous:** [mkboot](#)

*Das Linux Anwenderhandbuch*



## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# mkfs.ext2 (mke2fs)


## Funktion:

**mke2fs** erzeugt ein neues, erweitertes (ext2fs) Dateisystem auf einem formatierten Datenträger 

## Syntax:

**mke2fs** [-ctv] [-b *Größe*] [-f *Größe*] [-i *Anzahl*] [-l *Datei*] [-m *Verhältnis*] *Geräte*datei [*Größe*]

## Beschreibung:

Das **mke2fs**-Kommando legt auf einem Blockdevice, normalerweise einer Festplattenpartition, ein Dateisystem an. Für die Zusammenarbeit mit dem mkfs-Front-End muß das Programm (auch) unter dem Namen `mkfs.ext2` installiert sein.  In der Kommandobeschreibung steht `mke2fs` synonym für `mkfs.ext2`, das Verhalten des Programms ändert sich mit dem Namen nicht.

Die *Größe* der Partition oder Diskette muß nicht angegeben werden, `mke2fs` ermittelt diesen Wert selbständig.

Der Unterschied zum Minix-Dateisystem besteht für den Anwender vor allem darin, daß das ext2fs Partitionen bis zu 2 Gigabytes und Dateinamen mit einer Länge von bis zu 255 Zeichen zuläßt. Eine genauere Beschreibung des ext2fs ist im Anhang zu finden ([hier](#)).

In zukünftigen Versionen des erweiterten Dateisystems werden Blockgrößen von mehr als 1 Kilobyte wählbar sein; in der aktuellen Version enthält ein Block immer 1024 Bytes. Ebenso wird die geplante Fragmentierung der Blöcke noch nicht unterstützt.

## Optionen:

**-b *Größe***

(blocks) bestimmt die Größe der Blockgruppen

**-c**



(check) überprüft die Partition auf defekte Blöcke

**-f *Fragmentgröße***

bestimmt die *Fragmentgröße* zur Fragmentierung der Datenzonen (diese Option wird noch nicht unterstützt)

**-i *Anzahl***

setzt die Anzahl Bytes pro Inode fest; für jeweils *Anzahl* Bytes Plattenplatz wird eine Inode erzeugt (Mindestwert 1024, Voreinstellung 4096)

**-l *Datei***

liest die Nummern der defekten Blöcke aus der *Datei*

**-m *Anteil***

setzt den *Anteil* der Datenblöcke, die für den Superuser reserviert sind (Voreinstellung 5%); diese Einstellung kann auch nachträglich mit dem Programm `tune2fs` verändert werden

**-t**

(test) Überprüft die Partition auf defekte Blöcke

**-v**

(verbose) das Programm gibt sich informativ

**-F**

(Force) erzwingt die Ausführung von `mkfs`, auch wenn es sich bei dem Device um ein zeichenorientiertes Gerät handelt

**-S**

veranlaßt `mkfs` nur die Superblocks und Gruppendeskriptoren neu zu schreiben, die I-Nodes aber unangetastet zu belassen

**Autor:**

Theodore Ts'o

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [mkfs.minix \(mkfs\)](#) **Up:** [Die Kommandos für root](#) **Previous:** [mkfs \(Front-End\)](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

**Next:** [mkfs.xiafs \(mkxfs\)](#) **Up:** [Die Kommandos für root](#) **Previous:** [mkfs.ext2 \(mke2fs\)](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# mkfs.minix (mkfs)

## Funktion:

**mkfs.minix** erzeugt ein Dateisystem im Minix-Format auf einem formatierten Datenträger

## Syntax:

**mkfs.minix** [-c] [-l *Datei*] [-n*Zahl*] *Gerätedatei* *Größe*

## Beschreibung:

Mit Hilfe von **mkfs** wird auf leeren, formatierten Datenträgern (Disketten oder Festplatten) eine Struktur zur Verwaltung von Dateien und Verzeichnissen - das Minix-Dateisystem - angelegt.

Das Format des von **mkfs** angelegten Dateisystems ist identisch mit dem Dateisystem von A. Tanenbaum's Lehrbetriebssystem. Deshalb ist die Typenbezeichnung weiterhin `minix'. Eine genaue Beschreibung der Struktur dieses Dateisystems ist im [Anhang](#) zu finden.

Die *Gerätedatei* ist der Eintrag der entsprechenden Partition im Verzeichnis /dev. Der Name setzt sich aus der Bezeichnung für die gesamte (rohe) Festplatte (wie er beim [fdisk-Kommando](#) beschrieben ist) und der Partitionsnummer zusammen. Die zweite Partition der ersten AT-Bus-Platte wird beispielsweise als /dev/hda2 angesprochen; die erste Partition der zweiten SCSI-Festplatte heißt entsprechend /dev/sdb1.

Die *Größe* wird in Blöcken angegeben. Jeder Block ist ein Kilobyte groß, die genaue Zahl kann der von **fdisk** angezeigten Partitionstabelle entnommen werden.

Das **mkfs**-Kommando legt einen Superblock, Bitmaps für Inodes und Datenzonen und eine Anzahl Inodes an. Das Verhältnis von Dateisystemgröße zur Anzahl der Inodes kann bei **mkfs** nicht verändert werden.

Mit dem Minix-Dateisystem können Partitionen bis zu einer Größe von maximal 64 Megabyte verwaltet werden. Die Dateinamen in diesem Dateisystem können bis zu 14 Zeichen lang sein. Eine Unterteilung des Dateinamens in einen Stamm und eine Erweiterung, wie sie z. B. bei MS-DOS

erzwungen wird, gibt es nicht. Es steht dem Benutzer frei, beliebig viele Punkte in einem Dateinamen zu verwenden.

## Optionen:

-c

überprüft das Medium auf defekte Sektoren; jeder Block wird einmal beschrieben, wieder gelesen und verglichen; fehlerhafte Blöcke werden automatisch von einer `.badblocks`-Datei belegt

-l *Datei*

veranlaßt `mkfs.minix`, die Nummern der defekten Blöcke aus der *Datei* zu lesen

-n*Zahl*



stellt die maximale Länge für Dateinamen in dem erzeugten Dateisystem auf die angegebene Zahl; zulässige Werte sind nur 14 oder 30

## Autor:

Linus Torvalds

---

Next	Up	Previous	Contents	Index
------	----	----------	----------	-------

**Next:** [mkfs.xiafs \(mkxfs\)](#) **Up:** [Die Kommandos für root](#) **Previous:** [mkfs.ext2 \(mke2fs\)](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

**Next:** [mknod](#) **Up:** [Die Kommandos für root](#) **Previous:** [mkfs.minix \(mkfs\)](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
  - [Siehe auch:](#)
- 

# mkfs.xiafs (mkxfs)

## Funktion:

**mkxfs** legt ein xiafs auf einer Diskette oder Festplattenpartition an

## Syntax:

**mkxfs** [-c / -l *Datei*] [-k *Größe*] [-z *Größe*] *Gerät* *Größe*

## Beschreibung:

**mkxfs** legt auf einer formatierten Diskette oder auf einer Festplattenpartition *Gerät* ein xiafs (Xia-Filesystem) an. Die *Größe* muß in Kilobytes angegeben werden. Für die Zusammenarbeit mit dem neuen mkfs-Front-End muß das Programm (auch) unter dem Namen `mkfs.xiafs` installiert sein. In der Kommandobeschreibung steht `mkxfs` synonym für `mkfs.xiafs`, das Verhalten des Programms ändert sich mit dem Namen nicht.

Das Verhältnis von Inodes zu Datenzonen ist mit 1:4 unveränderlich festgelegt.

Eine genauere Beschreibung des xiafs ist im [Anhang](#) zu finden.

## Optionen:

-c

(check) sucht nach defekten Blöcken auf dem Datenträger und kennzeichnet sie automatisch als belegt

-l *Datei*

liest eine Liste defekter Blöcke aus der *Datei*, um sie als belegt zu markieren

-k *Größe*

reserviert einen Systembereich der angegebenen *Größe* für den Kernel; mit dem

mkboot-Kommando kann eine so eingerichtete xiafs-Partition bootfähig gemacht werden  
-z *Größe*  
richtet Datenzonen der angegebenen *Größe* ein (diese Option wird noch nicht unterstützt)

## Siehe auch:

[mkfs](#), [mke2fs](#), [mkboot](#), [xfscck](#)

## Autor:

Frank Q. Xia

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [mknod](#) **Up:** [Die Kommandos für root](#) **Previous:** [mkfs.minix \(mkfs\)](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# mknod

## Funktion:

**mknod** erzeugt eine Spezialdatei

## Syntax:

**mknod** [-m *Modus*] [-mode=*Modus*] *Name* {*b**c**u*} *Major Minor*

**mknod** [-m *Modus*] [-mode=*Modus*] *Pfad* *p*

## Beschreibung:

**mknod** erzeugt ein FIFO, eine Gerätedatei für ein zeichenorientiertes Gerät (character device) oder für ein blockorientiertes Gerät (block-device) mit dem angegebenen *Namen*.

Die Gerätedateien werden über die Major Device Nummern (Hauptgerätenummern) mit den entsprechenden Gerätetreibern im Kernel verbunden. Mehrere Geräte der gleichen Art werden vom Gerätetreiber durch die Minor Device Nummern (Untergerätenummern) unterschieden.

Eine vollständige Liste aller Gerätenummern finden Sie bei den Kernelsourcen in der Datei `./src/linux/Documentation/devices.txt`. Die verbindliche Liste aller registrierten Hauptgerätenummern befindet sich in der Includedatei `<linux/major.h>`.

Die Zugriffsrechte auf die Datei werden aus der Bitdifferenz von 0666 und der aktuellen *umask* des aufrufenden Prozesses gebildet.

Der erste Buchstabe nach dem *Namen* gibt den Typ der Datei an:

p

(pipe) erzeugt eine FIFO Spezialdatei (wie [mkfifo](#).)

b

(block) erzeugt eine Gerätedatei für ein (gepuffertes) blockorientiertes Gerät

c

(character) erzeugt eine ungepufferte Gerätedatei für ein zeichenorientiertes Gerät

u

(unbuffered) das Gleiche wie *c*

## Optionen:

-m *Modus*

setzt die Rechte der Dateien auf *Modus* wie bei [chmod](#)

## Autor:

David MacKenzie

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [mkswap](#) **Up:** [Die Kommandos für root](#) **Previous:** [mkfs.xiafs \(mkxfs\)](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Beispiel:](#)
  - [Optionen:](#)
- 

# mkswap

## Funktion:


**mkswap** bereitet eine (Geräte-) Datei für die Auslagerung von Prozessen vor

## Syntax:

**mkswap** [-c] *Datei Blöcke*

## Beschreibung:

In einem Multitasking-Betriebssystem wie Linux „schlafen“ fast immer einige Prozesse, weil sie auf das Ende eines anderen Prozesses warten. Daher ist es leicht möglich, bei großer Systembelastung schlafende Prozesse auf Festplatte auszulagern, um dadurch Platz im Arbeitsspeicher für neue Prozesse zu erhalten. Damit eine Partition entsprechend verwaltet werden kann, muß sie mit **mkswap** als Swapbereich eingerichtet werden. /dev/name ist die Gerätedatei zu der Partition, auf der der Swapbereich angelegt werden soll, und *Blöcke* ist die Größe in Blöcken. Um eine Swappartition zu aktivieren, muß das Kommando **swapon Swappartition** aufgerufen werden.

Es ist auch möglich, für kurzfristigen Speichermehrbedarf eine Swapdatei einzurichten. Dazu wird mit dem **dd**-Kommando eine leere Datei der benötigten Größe erzeugt und diese Datei dann mit dem **mkswap**-Kommando vorbereitet. 

## Beispiel:

```
# dd bs=1024 if=/dev/zero of=/dev/swapfile count=5120
5120+0 records in
5120+0 records out
# mkswap -c /dev/swapfile 5120
Setting up swapspace, size = 5238784 bytes
```



```
# sync
# swapon /dev/swapfile
# _
```

Diese Kommandofolge erzeugt eine Swapdatei von 5 Megabyte im Verzeichnis /dev.

Die Speicherverwaltung von Linux benutzt **paging**. Das heißt, es werden nicht sofort die kompletten Prozesse ausgelagert, sondern nur so viele Speicherseiten, wie ein anderer Prozeß gerade anfordert.

## Optionen:

-c  
(check) überprüft die Partition auf defekte Blöcke

## Autor:

Linus Torvalds

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [mount](#) **Up:** [Die Kommandos für root](#) **Previous:** [mknod](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

Next: [rdev](#) Up: [Die Kommandos für root](#) Previous: [mkswap](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# mount

## Funktion:

**mount** setzt das Dateisystem zusammen

## Syntax:

**mount** *[-afnrwuv] [-t Typ] [-o Schalter] [Gerätedatei] [Verzeichnis]*

## Beschreibung:

**mount** fügt die einzelnen Dateisysteme auf den verschiedenen Massenspeichern zu einem einzigen Dateisystembaum zusammen. Linux unterstützt verschiedene Arten von Dateisystemen, eine Liste finden Sie auf Seite [Dateisysteme](#).

Weitere Information finden im Kapitel über [Dateisysteme](#).

Die einzelnen Partitionen und Laufwerke sind als Gerätedateien (special-files) im Ordner `/dev` abgebildet. Die auf den Partitionen gespeicherten Dateisysteme werden durch den `mount` Befehl auf beliebige leere Verzeichnisse aufgesetzt.

`mount` erstellt eine Liste der aufgesetzten Dateisysteme in `/etc/mtab`. Der Kernel unterhält die gleiche Liste im Prozeßdateisystem in `/proc/mounts`. Wenn `mount` ohne Kommandozeilenargumente aufgerufen wird, zeigt es den Inhalt dieser Liste an. Wenn ein `mount`-Befehl unvollständig angegeben wird, werden die fehlenden Parameter automatisch aus der Datei `/etc/fstab` ergänzt.

Die Superuserin kann das Einbinden bestimmter Dateisysteme durch unprivilegierte Systembenutzer erlauben, indem für die entsprechenden Devices die Option `user` in der Datei `/etc/fstab` eingetragen wird.

Ältere Versionen von `mount` haben den Versuch, ein schreibgeschütztes Medium zum Lesen und Schreiben in das Dateisystem einzubinden abgebrochen, weil der Kernel ein solches mounten verweigert. Das aktuelle `mount` erkennt den Fehler und bindet das Dateisystem automatisch nur zum Lesen ein.

## Optionen:

-a

alle Dateisysteme werden automatisch wie in `/etc/fstab` beschrieben zusammengefügt

-f

(fake) führt alle Schritte des Befehls mit Ausnahme des eigentlichen Systemaufrufs aus; diese Option ist sinnvoll im Zusammenhang mit der Option -v

-n

unterdrückt den Eintrag des Dateisystems in die Datei `/etc/mtab` (nur `usermount`)

-o *Option* ...

bestimmt mit einer durch Komma getrennten Liste von *Optionen* verschiedene Eigenschaften des Dateisystems; eine Liste aller erlaubten Optionen finden Sie [hier](#)

-r

(read-only) veranlaßt das `mount`-Kommando, das Dateisystem Read-Only aufzusetzen; in so einem Dateisystem kann nicht geschrieben werden, auch wenn der Datenträger selbst nicht schreibgeschützt ist

-t *Typ*

spezifiziert den Typ des Dateisystems, das aufgesetzt werden soll; zusammen mit der -a Option können auch bestimmte Dateisystemtypen ausgeschlossen werden, indem anstelle des Typs die Angabe `noTyp` (z.B. `nomsdos` oder `nonfs`) gemacht wird

-u

(`usermount`) anstelle der Datei `/etc/fstab` wird `/etc/ufstab` benutzt; wenn das `usermount`-Kommando nicht von der Systemverwalterin aufgerufen wird, ist das die Voreinstellung

-v

(verbose) veranlaßt das `mount`-Kommando, Informationen zum Programmverlauf auf den Bildschirm zu schreiben

-w

(writable) veranlaßt das `mount`-Kommando, das Dateisystem mit Read-Write Erlaubnis aufzusetzen (nur bei Doug Quale's `mount`)

## Autor:

Doug Quale oder Peter Orbaek

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

Next: [rdev](#) Up: [Die Kommandos für root](#) Previous: [mkswap](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

**Next:** [rmmod](#) **Up:** [Die Kommandos für root](#) **Previous:** [mount](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# rdev

## Funktion:

**rdev** (root device) zeigt und verändert einige Kernelkonfigurationen

## Syntax:

**rdev** [-help] [-R *Image* {0,1}] [-r *Image* [*Größe*]] [-s *Image* [*Gerätedatei*]] [-v *Image* [*Modus*]] [*Image* [*Gerätedatei*]]

## Beschreibung:

**rdev** zeigt und verändert einige der Kernelkonfigurationen. Der Name ist von „root device“ abgeleitet und deutet auf die Hauptaufgabe des Programms hin, nämlich die Festplattenpartition mit dem Rootfilesystem neu festzulegen, ohne den Kernel neu zu übersetzen.

Die durch das **rdev**-Kommando veränderbaren Parameter werden in der Kerneldatei an einer bestimmten Stelle gespeichert. Die genaue Position kann durch einen *Offset* manipuliert werden. Die Voreinstellung ist 504 Bytes. In den auf diese Dateiposition folgenden Bytepaaren befinden sich die veränderbaren Parameter.

Die Kerneldatei kann sich entweder in einem Dateisystem befinden (zum Beispiel in der Datei `/Image`, `/zImage`, `/vmlinuz`, `/usr/src/linux/Image`), oder auf einer rohen Bootdiskette (zum Beispiel `/dev/fd0`)

## Optionen:

**-R *Image* {0,1}**

ändert das „Root-Flag“ im Kernelimage; wenn das Flag auf 1 gesetzt ist, wird das Root-Filesystem Read-Only aufgesetzt

**-s *Image* [*Gerätedatei*]**

legt die Swappartition im Kernel fest (wird seit der Kernelversion 0.98.3 nicht mehr unterstützt)

### **-v *Image* [*Modus*]**

zeigt oder setzt den Videomodus beim Systemstart; die folgenden Modi werden unterstützt:

-3

der Videomodus wird interaktiv erfragt (Voreinstellung)

-2

der erweiterte VGA-Modus mit 80x50 Zeichen

-1

der Standard-VGA Modus mit 80x25 Zeichen

1

der erste unterstützte Videomodus

2

der zweite unterstützte Videomodus (und so weiter)

Die unterstützten Videomodi hängen von der VGA Karte ab.

### **-r *Image* [*Key*]**

zeigt oder verändert die Einstellungen zum Laden einer RAM-Disk von Diskette. Die gleichen Einstellungen können auch durch die Kernelargumente `load_ramdisk`, `prompt_ramdisk` und `ramdisk_start` auf dem Bootprompt vorgenommen werden. Die Schlüsselzahl wird nach folgender Formel berechnet:

$$Key = ramdisk\_start + load\_ramdisk * 16384 + prompt\_ramdisk * 32768$$

Eine genauere Erklärung zu den Parametern finden Sie im Abschnitt über die [RAM-Disk](#).

Durch Links auf die Kommandonamen **ramsize** und **vidmode** können die Optionen `-r` beziehungsweise `-v` voreingestellt werden.

### **Autor:**

Werner Almesberger und Peter MacDonald

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [rmmod](#) **Up:** [Die Kommandos für root](#) **Previous:** [mount](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

Next: [rpm](#) Up: [Die Kommandos für root](#) Previous: [rdev](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Siehe auch:](#)
- 

# rmmod

## Funktion:

**rmmod** entfernt Laufzeitmodule aus dem Kernel

## Syntax:

**rmmod** [-r] *Modul*...

## Beschreibung:

**rmmod** versucht, die in der Kommandozeile benannten Module aus dem Kernel zu entfernen. Wenn auf der Kommandozeile mehrere Module angegeben sind, werden sie in der Reihenfolge ihres Auftretens entfernt.

Der Schalter *-r* veranlaßt **rmmod** einen ganzen Modulstack zu entfernen. Wenn so das oberste Modul eines Stacks gelöscht wird, werden alle Module die von diesem Modul benutzt wurden ebenfalls entfernt.

Es können nur unbenutzte, inaktive Module aus dem Kernel entfernt werden.

## Siehe auch:

[insmod](#) und das Kapitel über Module ab Seite 

## Autor:

Bas Larhoven, Jon Tombs und Bjorn Ekwall

**Next:** [shutdown](#) **Up:** [Die Kommandos für root](#) **Previous:** [rmmod](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Installation neuer Pakete](#)
  - [Upgrade existierender Software](#)
  - [Suche nach Informationen über ein Softwarepaket](#)
  - [Löschen einer installierten Software](#)
  - [Verifizieren der Integrität einer Softwareinstallation](#)
  - [Reparieren einer fehlerhaften Installation](#)
  - [Erzeugen neuer RPM-Pakete](#)
  - [Siehe auch:](#)
- 

# rpm

## Funktion:

**rpm** (RedHat Packet Manager) installiert und verwaltet Softwarepakete

## Syntax:

**rpm** *Optionen*

## Beschreibung:

Der RedHat Packet Manager **rpm** installiert neue Softwarepakete im RPM-Format auf einem Linux-System oder entfernt vorher installierte Pakete vollständig.

**rpm** stellt bei der Installation neuer Pakete die Integrität der bereits installierten Pakete sicher. **rpm** überprüft vor der Installation eines neuen Pakets die durch Abhängigkeit von anderen Paketen gegebenen Voraussetzungen. Beim Update eines installierten Pakets auf eine neuere Version werden die Konfigurationsdateien des alten Pakets automatisch gesichert.

Das Programm **glint** bietet eine komfortable Benutzerschnittstelle zu den wesentlichen Funktionen von **rpm**. Eine Einführung in die Benutzung von **rpm** auf der Kommandozeile ist im Abschnitt über [Softwaremanagement](#) zu finden.

Über die für die Systemverwaltung notwendige Funktionalität hinaus kann **rpm** dazu benutzt werden,

neue Softwarepakete im RPM-Format zu erzeugen.

Für die unterschiedlichen Aufgaben benutzt `rpm` jeweils bestimmte Betriebsarten. Die wichtigsten Betriebsarten sind:

## **Install**

Im Installationsmodus werden neue Softwarepakete aus einer RPM-Datei in das laufende System eingefügt. Der Installationsmodus wird durch den Optionsschalter `-i` oder in der langen Schreibweise `-install` eingeleitet. Mit zusätzlichen Optionsschaltern kann die Arbeitsweise von `rpm` im Installationsmodus weiter beeinflusst werden. Als Argument sind die Namen der RPM-Dateien für die zu installierenden Pakete anzugeben.

## **Upgrade**

Im Upgrade-Modus wird ein bereits installiertes Softwarepaket durch eine neuere Version ersetzt. Um `rpm` im Upgrade-Modus zu starten, muß als erste Option `-U` oder in langer Schreibweise `-upgrade` angegeben werden. Die Arbeitsweise gleicht in vieler Hinsicht der beim Installieren neuer Pakete. Entsprechend sind auch die möglichen Modifikationsoptionen und Argumente fast identisch mit denen des Installationsmodus.

## **Query**

Der Query-Modus liefert Informationen über die mit `rpm` bereits installierten Softwarepakete oder über den Inhalt einer RPM-Datei. `rpm` gelangt in den Query-Modus, wenn als erste Option ein `-q` oder `-query` angegeben wird.

## **Erase**

Der Erase-Modus dient zum vollständigen Löschen vorher installierter RPM-Pakete aus dem System. Der Modus wird durch die Schalter `-e` oder `-erase` eingeleitet. Konfigurationsdateien werden gesichert, sofern sie verändert wurden.

## **Verify**

Der Verify-Modus dient zum Überprüfen der Integrität bereits installierter Softwarepakete. Eingeleitet wird dieser Modus mit `-V` oder `-verify`. Verify erkennt beispielsweise das Fehlen von zu einem Softwarepaket gehörenden Dateien und die Veränderung von Eigentümer oder Zugriffsrechten.

## **Build**

Im Build-Modus werden neue RPM-Pakete erzeugt. Dieser Modus wird durch die Option `-b` eingeleitet.

Die Optionen `-help` `-version` `-showrc` `-rebuilddb` `-initdb` `-recompile` `-rebuild` `-setperm` `-setugids` und `-checksig` können unabhängig von einem der oben beschriebenen Modi verwendet werden.

Um während der Ausführung eines `rpm`-Kommandos zusätzliche Information über den Status und eventuell hilfreiche Debuginformationen zu erhalten, kann in jedem Modus durch die Option `-v` oder `-v v` die Gesprächigkeit des Programms erhöht werden.

Die Optionen `-help` und `-version` führen zur Ausgabe der entsprechenden Informationen.

Verschiedene Pfadinformationen und bestimmte Defaulteinstellungen kann `rpm` aus einer



Konfigurationsdatei lesen. Das Programm sucht nach dieser Datei unter dem Namen `rpmrc` in den Verzeichnissen `/usr/lib` und `/etc`. Außerdem werden die Einstellungen aus der Datei `.rpmrc` im Heimatverzeichnis des Benutzers gelesen, wenn diese Datei existiert. Schließlich ist es noch möglich, mit der Option `-rcfile` eine Initialisierungsdatei beim Aufruf des Programms auf der Kommandozeile anzugeben. Es werden alle Dateien in der oben aufgeführten Reihenfolge gelesen und eventuell bereits belegte Werte durch die später gelesenen überschrieben.

## Installation neuer Pakete

Der sicher am häufigsten verwendete Modus von `rpm` ist der Installationsmodus.

Als Argument erwartet das Programm den oder die Namen der zu installierenden RPM-Datei(en). Wenn diese Dateien auf CD, in einem NFS-Verzeichnis oder auf der lokalen Festplatte vorhanden sind, wird das Paket durch seinen normalen Linux-Dateinamen angegeben. `rpm` erlaubt aber auch die direkte Installation von RPM-Paketen von einem FTP-Server. Dazu muß einfach nur der Paketname als URL (das ist die in HTML-Dokumenten für das WWW übliche Form) angegeben werden. Das Beispiel zeigt eine URL für die Datei `bar.rpm` im Verzeichnis `/pub/foo` auf dem FTP-Server `server.domain.de`. Wenn ein Username und ein Paßwort erforderlich sind, können die ebenfalls angegeben werden. Ein anonymer FTP-Login erfolgt automatisch.

```
ftp://user:password@server.domain.de/pub/foo/bar.rpm
```

Zusätzlich zu dem immer erforderlichen Optionsschalter `-i` (bzw. der langen Version `-install`) bietet der Installationsmodus folgende Optionen:

`-test`

führt alle normalerweise vor der eigentlichen Installation vorgenommenen Tests durch, die Installation selbst findet aber nicht statt. Getestet wird, ob alle Pakete, von denen ein zu installierendes Paket abhängig ist, bereits installiert sind und ob eine der bei der Installation erzeugten Dateien einen Namenskonflikt mit einer bereits installierten Datei eines anderen Pakets auslöst.

Wird bei einem der Tests ein Problem erkannt, erscheint eine entsprechende Fehlermeldung und das Programm beendet mit dem Status 1. Anderenfalls wird das Programm ohne weitere Ausgabe mit dem Status 0 beendet.

`-replacepks`

erzwingt die Installation eines bereits installierten Pakets. `rpm` erkennt durch den Vergleich mit seiner Datenbank, daß ein Paket bereits installiert ist, und lehnt die Wiederinstallation normalerweise mit einer Fehlermeldung ab. Die Durchführung einer solchen Wiederinstallation kann durch diese Option veranlaßt werden. Die existierenden Dateien werden überschrieben, Konfigurationsdateien aber vorher gesichert. Mit dieser Option kann ein teilweise zerstörtes Paket wieder vervollständigt werden.

`-replacefiles`

veranlaßt das Überschreiben bereits installierter Dateien anderer Pakete. `rpm` sichert die Integrität der bereits installierten Softwarepakete, indem es das Überschreiben existierender Dateien durch Dateien gleichen Namens, aber anderen Inhalts aus anderen Paketen verhindert.

Konfigurationsdateien werden vor dem Überschreiben automatisch gesichert, wenn sie beim Erzeugen der RPM-Datei als solche gekennzeichnet wurden.

## `-nodeps`

unterdrückt die Prüfung auf nicht erfüllte Abhängigkeitsbedingungen zu anderen Paketen. Abhängigkeiten von ``Services''(beispielsweise eine Shared Library) anderer Pakete sind in der RPM-Datei verzeichnet. `rpm` stellt anhand seiner Datenbank fest, ob eines der bereits installierten Pakete die Abhängigkeit auflöst. Es lehnt die Installation der neuen Software ab, wenn nicht alle Abhängigkeiten aufgelöst werden können. Um die Installation trotzdem durchzuführen, muß diese Option verwendet werden.

In bestimmten Fällen kann anstelle einer Abhängigkeit zu einem anderen Paket auch ein ausschließender Konflikt zu einer anderen Software in einem RPM-Paket verzeichnet sein. Auch in diesem Fall kann durch diese Option die Installation des neuen Pakets erzwungen werden.

## `-force`

entspricht der Kombination von `-replacepkgs` und `-replacefiles`. Lediglich die Abhängigkeit von anderen Paketen wird bei `-force` weiterhin getestet.

## `-excludedocs`

unterdrückt die Installation der zu einem Softwarepaket gehörenden Dokumentationsdateien.

Um die Installation von Dokumentation auf dem lokalen Rechner generell zu verhindern, kann in der Datei `/etc/rpmrc` (oder einer der anderen oben genannten Initialisierungsdateien) die Zeile ```excludedocs: 1`'' eingetragen werden.

## `-includedocs`

veranlaßt die Installation von Dokumentationsdateien, obwohl das in der Initialisierungsdatei durch ```excludedocs: 1`'' anders voreingestellt wurde.

## `-hash`

führt zur Ausgabe von 50 ``Hashmarks'' (#) während der Installation der Daten. Durch diese Anzeige wird der Fortschritt bei der laufenden Installation sichtbar. Durch Hinzufügen der zusätzlichen Option `-v` entsteht bei der Installation mehrerer Pakete eine übersichtliche Anzeige.

## `-ftpproxy Server`

Wenn der Name des zu installierenden RPM-Pakets als URL angegeben ist, das Paket also direkt per FTP installiert werden soll, kann durch diese Option der eventuell erforderliche Name des FTP-Proxy-Servers angegeben werden.

## `-ftpport Port`

veranlaßt den Aufbau einer FTP-Verbindung auf dem angegebenen *port*.

## `-noscripts`

unterdrückt die Ausführung von Prä- und Post-Install-Scripts bei der Installation des Pakets. Die Ausführung dieser Scripts ist in der Regel für die vollständige Installation und für das einwandfreie Funktionieren eines Softwarepakets notwendig. Diese Option ist deshalb nur beim Debuggen eigener RPM-Pakete sinnvoll.

## `-prefix Verzeichnis`

veranlaßt die Installation eines verschiebbaren Pakets in dem angegebenen *Verzeichnis*. Mit dem Kommando

```
rpm -qp --queryformat "%{defaultprefix}\n" *.rpm
```

kann festgestellt werden, ob ein bestimmtes Paket verschoben werden kann. Die Ausgabe

“(none)” zeigt an, daß ein Paket **nicht** verschiebbar ist. Anderenfalls wird der Defaultprefix durch den mit dieser Option angegebenen Wert ersetzt.

-ignorearch

erzwingt die Installation eines Pakets auch dann, wenn es für eine andere Rechnerarchitektur erzeugt wurde.

-ignoreos

erzwingt die Installation eines Pakets auch dann, wenn es für ein anderes Betriebssystem erzeugt wurde.

-root *Verzeichnis*

macht das angegebene *Verzeichnis* zum Wurzelverzeichnis der Installation. Durch diese Option wird das gesamte Installationssystem in das *Verzeichnis* verschoben. Normalerweise geschieht das nur bei der Erstinstallation eines Systems.

-dbpath *Verzeichnis*

veranlaßt die Benutzung der RPM-Datenbank aus dem angegebenen *Verzeichnis*. Normalerweise befindet sich die Datenbank in `/var/lib/rpm`.

-percent

veranlaßt die Ausgabe einer prozentualen Fortschrittsanzeige zur Auswertung durch ein anderes Programm während der laufenden Installation.

## Upgrade existierender Software

Das Upgrade eines bereits installierten Softwarepakets auf eine neuere Version entspricht einer Kombination aus Installation der neuen und Löschung der alten Dateien. Die Herausforderung an rpm bei der Automatisierung dieses Vorgangs besteht in der Bewahrung eventuell vorhandener Konfigurationsdateien mit systemspezifischen Einstellungen.

rpm löst das Problem, indem es die existierenden Konfigurationsdateien sichert, wann immer das nötig ist. Damit das Programm die Sicherung durchführen kann, müssen zunächst die Konfigurationsdateien als solche gekennzeichnet sein. Diese Kennzeichnung ist Bestandteil der RPM-Datei. Es ist also in der Verantwortung der Distributoren oder der Entwickler, die Konfigurationsdateien als solche zu markieren.

Beim Upgrade werden existierende Konfigurationsdateien gesichert und durch die Endung `.rpmsave` erweitert. Wenn die originale Konfigurationsdatei der alten Softwareversion mit der zu installierenden Konfigurationsdatei der neuen Softwareversion identisch ist, wird die existierende Konfigurationsdatei übernommen, wenn sie sich von der originalen unterscheidet.

rpm verfügt nicht über die Möglichkeit, inhaltliche Übereinstimmung festzustellen oder gar Konfigurationsdateien zu konvertieren. Zur Überprüfung der Identitäten benutzt das Programm sogenannte MD5 Prüfsummen. Diese Prüfsummen werden über die vollständigen Dateien erzeugt und von rpm bei der Installation eines Pakets in seiner Datenbank gespeichert. Das Prüfsummenverfahren ist ein sehr zuverlässiger Weg zur Unterscheidung unterschiedlicher Versionen einer Datei. Durch die in der Datenbank gespeicherte Prüfsumme der originalen Datei kann rpm jederzeit Veränderungen an den installierten Daten feststellen.

Der Upgrade-Modus muß durch die Option `-U` (bzw. das äquivalente `-upgrade`) eingeleitet werden. Alle für den Installationsmodus beschriebenen Optionen können auch für den Upgrade-Modus

verwendet werden.

Zusätzlich versteht `rpm` im Upgrade-Modus eine weitere Option:

`-oldpackage`

erlaubt das "Downgrade" zu einer älteren Version eines installierten Softwarepakets.

Normalerweise erkennt das Programm den Versuch eines solchen "Rückschritts" als Fehler.

Diese Option ermöglicht es, nach dem Test der leider fehlerhaften aktuellsten Version einer Software wieder zum funktionierenden Ausgangszustand zurückzukehren.

## Suche nach Informationen über ein Softwarepaket

Beim Softwaremanagement geht es nicht nur darum, neue Programme zu installieren und zu aktualisieren. Viele Programme müssen konfiguriert werden, in jedem Fall muß die Dokumentation gelesen werden, eventuell müssen Fehler und Integrationsprobleme gelöst werden. `rpm` unterstützt Systemverwalterin und Benutzer, indem es einen einfachen Zugang zu den in seiner Datenbank gespeicherten Informationen anbietet.

Im Query-Modus kann `rpm` beispielsweise herausfinden, zu welchem Paket eine bestimmte Datei gehört, wo und unter welchem Namen die Konfigurationsdateien oder die Dokumentation installiert wurden und welche "Services" von einem Paket benutzt oder bereitgestellt werden.

Damit `rpm` im Query-Modus arbeitet, muß als erste Option `-q` (oder als Alternative `-query`) angegeben werden. Welche Information aus welchen Paketen gesucht wird, kann durch weitere Optionen festgelegt werden.

In der Regel werden die im Query-Modus ermittelten Informationen nicht über alle installierten Pakete gleichzeitig benötigt. `rpm` ermöglicht deshalb die Auswahl der zu untersuchenden Pakete nach verschiedenen Kriterien.

Die einfachste Methode zur Spezifikation eines zu untersuchenden Pakets ist natürlich die Übergabe des Paketnamens als Argument auf der Kommandozeile. Hierbei ist allerdings darauf zu achten, daß die Paketnamen exakt und vollständig angegeben werden müssen. Wildcards sind ebenso wenig erlaubt wie die Ignorierung von Groß- und Kleinschreibung. Zusätzlich zu dem eigentlichen Paketnamen kann noch die Versionsnummer und die Releasenummer des Pakets in der exakten Form mitgegeben werden.

Der exakte Paketname, von dem nur ein ungenauer Teil bekannt ist, kann mit Hilfe von `grep` aus der Liste aller installierten Pakete herausgefiltert werden:

```
$ rpm -q -a | grep file
fileutils-3.12-3
file-3.19-2
$ _
```

Weitere Möglichkeiten zur Auswahl bestimmter Pakete werden durch folgende Optionen gegeben:

`-a`

führt das Query-Kommando für alle in der Datenbank eingetragenen Softwarepakete durch.

`-p RPM-Datei`

untersucht die angegebene *RPM-Datei*, nicht die Datenbank des bereits installierten Systems. Mit dieser Option kann der Inhalt eines unbekannten RPM-Pakets untersucht werden, bevor die darin enthaltene Software installiert wird.

**-f *Datei***

ermittelt anhand der Datenbank, zu welchem Paket die angegebene Datei gehört und gibt die Information zu diesem Paket aus.

**-whatprovides *Service***

ermittelt anhand der Datenbank, welches Softwarepaket den angegebenen Service (beispielsweise eine Shared Library) anbietet und gibt die Information zu diesem Paket aus.

**-whatrequires *Service***

ermittelt anhand der Datenbank, welche Softwarepakete den angegebenen Service verwenden und gibt Informationen zu diesen Paketen aus.

**-g *Gruppe***

ermittelt anhand der Datenbank alle zur angegebenen Gruppe gehörenden Softwarepakete und führt das Query-Kommando mit diesen Paketen aus. Die Gruppierung der einzelnen Pakete liegt ebenso wie die Bezeichnung der Gruppen im Ermessen des Distributors. Eine Liste mit allen verwendeten Programmgruppen wird durch das folgende Kommando ausgegeben: `rpm -q --qf '%{group}\n' -a | sort | uniq`

Ohne spezielle Auswahl zeigt `rpm` nur den Paketnamen (das ``Label'') an. Optional können folgende Informationen über die von `rpm` verwalteten Softwarepakete gewonnen werden:

**-i**

gibt die ``Karteikarte" zu einem Paket aus. Unter anderem sind darauf die Herkunft (Distribution), die Versionsnummer, das Installationsdatum und eine verbale Beschreibung des Inhalts verzeichnet.

Das Ausgabeformat und der Inhalt der Anzeige kann mit der Option `-queryformat` im Detail eingestellt werden.

**-l**

zeigt ein Listing aller zu einem Paket gehörenden Dateien. Durch Zusatz der Option `-v` entsteht ein ausführliches Listing im Format von `'ls -l'`.

**-c**

gibt eine Liste aller zu dem Paket gehörenden Konfigurationsdateien aus. Wenn die Software keine Konfigurationsdatei benutzt, wird kein Dateiname ausgegeben.

**-d**

zeigt eine Liste aller zu einem Paket gehörenden Dokumentationsdateien an.

**-s**

zeigt die in der Datenbank gespeicherten Statusinformationen der zu einem Paket gehörenden Dateien an. Es findet keine Überprüfung im Dateisystem statt, die angezeigten Werte stammen allein aus den Datenbankeinträgen, die bei der Installation des Softwarepakets angelegt worden sind. Der Status kann folgende Werte annehmen:

**normal**

Die Datei wurde nicht durch die Installation eines anderen Pakets verändert.

## replaced

Die Datei wurde bei der Installation eines anderen Pakets ersetzt.

## not installed

Die Datei wurde nicht installiert. Dieser Fall tritt beispielsweise auf, wenn durch die Option `-excludedocs` ein Teil der zu einem Paket gehörenden Dateien ausgeschlossen wurde.

## net shared

Die Datei befindet sich in einem verteilt genutzten Netzwerkverzeichnis. Die Pfade solcher per NFS gemounteten oder exportierten Verzeichnisse müssen vor der Installation in der Initialisierungsdatei `/etc/rpmrc` in dem Eintrag `netsharedpath` angegeben sein, damit der Status entsprechend in der Datenbank festgehalten wird.

## `-queryformat` *FormatString*

gibt eine "Karteikarte" für jedes untersuchte Paket in dem durch den *FormatString* festgelegten, benutzerspezifischen Format aus. Diese Option kann auch durch `-qf` abgekürzt werden.

Der Formatstring wird ähnlich aufgebaut wie der bei der C-Funktion `printf` typischerweise verwendete. In einen Text können *Tags* eingebettet werden, die bei der Anzeige durch entsprechend formatierte Paketinformationen ersetzt werden. Ein Tag hat die Form `%{TAGNAME}`. Eine Liste aller möglichen Tagnamen wird durch die Option `-querytags` ausgegeben.

Zwischen dem Prozentzeichen und dem Tagnamen kann zur Ausrichtung einer Tabelle noch die Feldlänge angegeben werden. Soll eine Liste gleichartiger Einträge ausgegeben werden (beispielsweise eine Dateiliste), so muß der zu wiederholende Teil des FormatStrings in eckige Klammern eingeschlossen werden. Soll ein gleichbleibender Tag in mehreren Ausgabezeilen erscheinen, muß der Tagname durch ein Gleichheitszeichen gekennzeichnet werden. Das folgende Beispiel zeigt einen Formatstring zur Ausgabe des Paketnamens vor jeder Datei einer Liste:

```
$ rpm -q --qf '[%=name]: %-40{filenames} (%{filesizes} bytes)\n' ical
ical: /usr/bin/ical (10 bytes)
ical: /usr/bin/ical-2.0p2 (778268 bytes)
ical: /usr/doc/ical-2.0p2-5 (1024 bytes)
ical: /usr/doc/ical-2.0p2-5/ical.doc (54451 bytes)
....
$ _
```

Die Option `-queryformat` gibt eine vollständige Dateiliste aus, wie `-l`. Die Einschränkung der Liste, beispielsweise auf die Konfigurationsdateien, ist nicht möglich.

## `-querytags`

gibt eine Liste mit allen vom rpm Queryformat unterstützten Tags aus.

## `-provides`

veranlaßt die Anzeige aller von einem Paket bereitgestellten Services. Diese Services können von anderen Paketen genutzt werden.

## `-requires`

veranlaßt die Anzeige aller von einem Paket genutzten Services. Alle angeforderten Services müssen von bereits installierten Paketen angeboten werden, damit das Softwarepaket voll

funktionsfähig ist.

#### **-scripts**

gibt alle zu einem Paket gehörenden Scriptdaten aus. Das RPM-System benutzt an fünf verschiedenen Stellen paketspezifische Shellscripsts:

#### **preinstall**

wird vor der Installation des Pakets ausgeführt

#### **postinstall**

wird im Anschluß an die erfolgreiche Installation der Software ausgeführt

#### **preuninstall**

wird vor dem Entfernen eines Softwarepakets ausgeführt

#### **postuninstall**

wird nach der Löschung eines Softwarepakets ausgeführt

#### **verify**

wird zur Überprüfung einer fehlerfreien Installation im Verify-Modus ausgeführt.

#### **-dump**

gibt in Verbindung mit `-l`, `-d` oder `-c` alle zu einer Datei in der Datenbank enthaltenen Informationen aus. Das Format ist eher für die Interpretation durch andere Programme als für den menschlichen Leser gedacht. Über die bei einem ausführlichen Listing gelieferten Daten hinaus wird bei `-dump` auch die MD5 Prüfsumme der Datei angezeigt.

## **Löschen einer installierten Software**

Beim Löschen eines installiertes Softwarepakets "von Hand" können zwei Probleme auftreten:

1.

Es ist nicht ohne weiteres feststellbar, welche Dateien zu dem zu löschenden Paket gehören und welche mit anderen Paketen installiert oder gar während des Betriebs erzeugt wurden. Deshalb bleiben entweder zum Paket gehörende Dateien ohne weitere Funktion zurück oder es werden versehentlich Dateien anderer Softwarepakete gelöscht.

2.

Eventuell werden bestimmte Teile des zu löschenden Pakets von anderen Softwarepaketen verwendet, die nach dem Löschen dann nicht mehr funktionieren. Diese Abhängigkeiten sind ebenfalls nicht ohne weiteres feststellbar.

Beide Probleme lassen sich zuverlässig lösen, indem ganze Softwarepakete mit `rpm` gelöscht werden. `rpm` behält alle Informationen über ein mit dem Programm installiertes Softwarepaket in seiner Datenbank und kann so leicht alle zu dem Paket gehörenden Dateien identifizieren. Zusätzlich sind in der Datenbank die Abhängigkeiten der Pakete untereinander verzeichnet. Mit diesen Informationen verhindert `rpm` automatisch, daß Pakete gelöscht werden, die noch von anderen Paketen gebraucht werden.

`rpm` sichert zusätzlich die Konfigurationsdateien, sofern diese nicht mit den ursprünglich installierten Defaultdateien übereinstimmen.

Durch spezielle Shellscripsts, die von den Herstellern in ein RPM-Paket eingebaut werden können, lassen sich zusätzliche Kommandofolgen zur Vorbereitung und zum Abschluß einer Löschung

automatisch ausführen.

Der Erase-Modus muß durch die Option `-e` oder in ausgeschriebener Form `-erase` eingeleitet werden. Durch Angabe der zusätzlichen Option `-v` `v` wird ein ausführliches Protokoll des Löschvorgangs ausgegeben. Außerdem stehen folgende Optionen im Erase-Modus zur Verfügung:

`-test`

führt lediglich den Abhängigkeitstest durch, ohne das Paket tatsächlich zu löschen. Eine detaillierte Protokollierung aller für die Löschung vorgesehenen Schritte wird durch die zusätzliche Angabe der Doppeloption `-v` `v` ausgegeben.

Bei einem Testlauf werden auch die an der Löschung beteiligten Scriptdateien getestet, das bedeutet, sie werden ausgeführt! Die Ausführung kann verhindert werden, indem `-test` zusammen mit `-noscripts` aufgerufen wird. Der Inhalt der `uninstall`-Scripts kann im Query-Modus durch die Option `-scripts` angezeigt werden.

`-noscripts`

unterdrückt die Ausführung der `uninstall`-Scripts.

`-nodeps`

erzwingt das Löschen eines Softwarepakets auch dann, wenn die von dem Paket bereitgestellten Services von anderen Programmen verwendet werden. Nach einer solchen erzwungenen Löschung muß damit gerechnet werden, daß die davon abhängigen Softwarepakete nicht mehr einwandfrei funktionieren.

## Verifizieren der Integrität einer Softwareinstallation

Gelegentlich taucht die Frage nach der Integrität eines Softwarepakets auf. Das kann bei der Suche nach der Ursache für einen plötzlich auftretenden Fehler sein, oder es geschieht, nachdem eine Person unberechtigten Zugang zum Rechner hatte.

`rpm` kann mit Hilfe seiner Datenbank auch hier sehr gute Dienste leisten. Im `Verify`-Modus prüft `rpm` alle zu einem Paket gehörenden Dateien im Dateisystem und vergleicht die Daten mit den in der Datenbank gespeicherten. Alle Abweichungen werden als Fehlermeldungen angezeigt. Außerdem wird geprüft, ob alle Services, von denen das Paket abhängig ist, durch andere Pakete bereitgestellt werden.

Wenn `rpm` im `Verify`-Modus einen Fehler erkennt, gibt es den Namen der veränderten Datei zusammen mit einer Reihe von 8 oder 9 Zeichen aus. Die 8 Zeichen repräsentieren die 8 Tests, die im `Verify`-Modus durchgeführt werden. Das neunte Zeichen kennzeichnet Konfigurationsdateien.

Die einzelnen Zeichen und die damit verbundenen Tests sind folgende:

**S**

(Size) erscheint, wenn die Größe der Datei im Dateisystem von dem in der Datenbank gespeicherten Wert abweicht.

**M**

(Mode) erscheint, wenn die Zugriffsrechte der Datei im Dateisystem anders gesetzt sind, als das in der Datenbank verzeichnet ist.

**5**



(md5) erscheint, wenn die MD5-Prüfsumme für die Datei im Dateisystem nicht mit der in der Datenbank gespeicherten übereinstimmt.

## D

(Device) erscheint bei Gerätedateien, wenn die Gerätenummern der Datei im Dateisystem nicht mit den in der Datenbank gespeicherten Werten übereinstimmt.

## L

(Link) erscheint bei einem symbolischen Link, wenn der darin verzeichnete Dateiname nicht mit dem in der Datenbank gespeicherten übereinstimmt.

## U

(User) erscheint, wenn der Eigentümer einer Datei im Dateisystem nicht mit dem in der Datenbank verzeichneten Eigentümer übereinstimmt.

## G

(Group) erscheint, wenn die Gruppenzugehörigkeit einer Datei nicht mit der in der Datenbank verzeichneten Gruppe übereinstimmt.

## T

(Time) erscheint, wenn die Zeit der letzten Änderung einer Datei im Dateisystem nicht mit der in der Datenbank verzeichneten Zeit übereinstimmt.

Die Zeichen erscheinen, wenn der entsprechende Test **nicht** bestanden wurde. Für jeden bestandenen Test erscheint ein einzelner Punkt.

Durch den zusätzlichen Buchstaben c werden Konfigurationsdateien gekennzeichnet.

Der Verify-Modus von rpm muß durch die Option -V oder durch die gleichwertigen Optionen -y oder -verify eingeleitet werden. Als Argument kann der Paketname auf der Kommandozeile übergeben werden. Wie beim Query-Modus beschrieben, muß dieser Name vollständig und exakt angegeben werden.

Durch die Angabe der Doppeloption -v v auf der Kommandozeile erscheinen während der Ausführung zusätzliche Informationen über den Programmablauf. Außerdem stehen folgende Kommandozeilenoptionen zur Verfügung:

-p *RPM-Datei*

führt die Prüfung der Dateien im Dateisystem nicht gegen die in der Datenbank gespeicherten Informationen durch, sondern gegen die in der angegebenen *RPM-Datei* enthaltenen.

-f *Datei*

veranlaßt die Prüfung des gesamten Pakets, zu dem die angegebene *Datei* gehört.

-a

veranlaßt die Prüfung sämtlicher in der Datenbank verzeichneten Softwarepakete.

-g *Programmgruppe*

veranlaßt die Prüfung aller Pakete aus der angegebenen *Programmgruppe*.

-noscripts

unterdrückt die Ausführung des mit dem RPM-Paket ausgelieferten Verifikations-Scripts.

-nodeps

unterdrückt die Prüfung auf nicht aufgelöste Abhängigkeiten zu anderen Paketen.

`-nomd5`

unterdrückt den Test auf unterschiedliche MD5-Prüfsummen.

`-nofiles`

unterdrückt die Durchführung aller dateispezifischen Tests. Lediglich das Verify-Script und der Abhängigkeitstest werden ausgeführt.

Weil sehr viel Linux-Software über das Internet verteilt wird, besteht Bedarf an einer weiteren Art der Software-Überprüfung, nämlich der Verifizierung des RPM-Pakets selbst. Es gibt zwar praktisch keine Viren unter Linux; Trojanische Pferde oder einfache ``Paketbomben" können aber leicht in RPM-Dateien versteckt und über das Internet verbreitet werden. Um solchen Mißbrauch zuverlässig zu verhindern, kann die Integrität der RPM-Datei mit Hilfe einer PGP-Signatur zweifelsfrei festgestellt werden.

PGP (*Pretty Good Privacy*) ist ein sehr leistungsfähiges Programm zum Verschlüsseln von Daten und zum Signieren von Dateien. Es arbeitet mit einem kryptografisch starken Public Key Verfahren. Einige Hersteller von Linux-Distributionen und viele Entwickler signieren ihre Produkte mit einem nur ihnen bekannten geheimen Schlüssel. Der dazu passende öffentliche Schlüssel wird auf verschiedenen Wegen verbreitet, beispielsweise auf den WEB-Servern der Anbieter oder über öffentliche Key-Server.

Um die PGP-Signatur eines Pakets prüfen zu können, muß die PGP-Software installiert sein und der Public Key des Paketherstellers am PGP-Schlüsselring hängen.

`-checksig RPM-Datei`

führt die Signaturprüfung für die angegebene *RPM-Datei* durch.

Nach der erfolgreichen Durchführung der Prüfung wird eine Zeile mit dem Paketenamen und den Wörtern ``size pgp md5 OK" ausgegeben. Wenn das Paket keine PGP-Signatur hat, erscheint nur ``size md5 OK". Ein defektes oder manipuliertes Paket wird durch die Ausgabe ``size PGP MD5 NOT OK" gekennzeichnet.

## Reparieren einer fehlerhaften Installation

Die Feststellung eines Fehlers durch `rpm` im Verify-Modus ist nur der erste Schritt zur Lösung des Problems. Wenn das Fehlen von Dateien oder eine nicht beabsichtigte Veränderung von Daten oder Programmen festgestellt wurde, bleibt eigentlich nur das Überschreiben der betroffenen Softwarepakete. `rpm` ermöglicht das mit der Option `-replacepkgs` im Installationsmodus.

Manchmal ist es jedoch gar nicht nötig, die Daten zu überschreiben. Viele Probleme mit einer Software rühren von falschen Eigentums- und Zugriffsrechten her. Die korrekten Voreinstellungen für diese Dateiattribute sind in der Datenbank von `rpm` verzeichnet und können deshalb jederzeit in ihren ursprünglichen Zustand zurückgesetzt werden:

`-setperms`

setzt die Zugriffsrechte für alle Dateien des spezifizierten Pakets auf die in der Datenbank gespeicherten Werte.

`-setugids`

setzt Eigentümer und Gruppe aller spezifizierten Pakete auf die in der Datenbank gespeicherten Werte.

Für beide Optionen können die zu bearbeitenden Softwarepakete auf die beim Query-Modus

beschriebene Weise angegeben werden.

Sollte es aus irgend einem Grund zu einem Fehler in der Datenbank von `rpm` selbst kommen, ist die korrekte Verwaltung der Softwarepakete im System nicht mehr möglich. Bevor diese Software selbst neu installiert wird und damit die Datenbank mit allen Informationen über das bestehende System verloren geht, kann mit der Option `-rebuilddb` eine Rekonstruktion der Datenbank aus eventuell noch vorhandenen Resten versucht werden.

Eine vollständig zerstörte Datenbank kann aus dem laufenden System nicht mehr zurückgewonnen werden. Mit der Option `-initdb` kann eine neue, leere Datenbank erzeugt werden, nachdem die eventuell noch vorhandenen Reste einer nicht mehr brauchbaren Datenbank gelöscht wurden.

## Erzeugen neuer RPM-Pakete

Die Funktionalität von `rpm` wird abgerundet durch den Build-Modus, in dem neue RPM-Pakete erzeugt werden. Die Herstellung von RPM-Paketen ist nur auf den ersten Blick allein Aufgabe der Linux-Distributoren. Wegen der vielen Vorteile, die das Softwaremanagement mit `rpm` bietet, ist es durchaus auch für eine Systemverwalterin interessant, lokal aus Sourcecode erzeugte Software mit `rpm` zu installieren.

`rpm` erledigt wesentliche Teile der Herstellungsarbeit automatisch, so daß der Aufwand für die Umwandlung eines TAR-Pakets in eine RPM-Datei überschaubar bleibt. In der Hauptsache besteht die manuelle Arbeit darin, das sogenannte Spec-File zu erzeugen. Ähnlich dem Makefile steuert diese Datei die Aktionen von `rpm`.

Die Herstellung eines RPM-Pakets geschieht in mehreren Phasen:

### **prep**

Die Vorbereitung kann beispielsweise im Auspacken der Sourcen, dem Anwenden eines Patches auf diese Sourcen und dem Ausführen von `configure` bestehen.

### **build**

Der Aufbau besteht in vielen Fällen einfach aus der Ausführung von `make` im Source-Verzeichnis.

### **install**

Die Erstinstallation wird ebenfalls meistens von `make` erledigt

### **binary**

Die neu installierten Dateien werden der Packliste entsprechend in einer RPM-Datei zusammengefaßt.

### **archive**

Schließlich werden die Sourcen zusammen mit dem Spec-File, den Patches und allen eventuell sonst noch für die Herstellung nötigen Dateien in ein Source-RPM gepackt.

Zur Unterstützung bei der Erstellung der Packliste kann `rpm` im Build-Modus prüfen, ob alle Dateien der Liste vorhanden sind und zusätzlich bestimmte Services und Abhängigkeiten feststellen.

Der Build-Modus muß durch die Option `-b` eingeleitet werden. Direkt an diesen Optionsschalter angehängt wird ein zweiter Buchstabe, mit dem festgelegt wird, welche Herstellungsphasen durchlaufen werden sollen:

p

nur die Vorbereitungsphase

c

die Vorbereitung und der Aufbau

i

Vorbereitung, Aufbau und Erstinstallation

b

wie *i*, nach der Erstinstallation wird das binäre RPM-Paket gepackt

a

wie *b*, zusätzlich wird noch das Source-RPM erzeugt

l

Überprüfung der Packliste

Zusätzlich stehen im Build-Modus folgende Optionen zur Verfügung:

`-short-circuit`

veranlaßt bei den Modi `-bc` und `-bi` nur die Ausführung der jeweils letzten Phase. Auf diese Weise kann ein wegen eines Fehlers unterbrochener Herstellungsvorgang wieder aufgenommen werden.

`-test`

sichert die von `rpm` erzeugten Shellscripts in `/var/tmp`. Diese Dateien enthalten alle während des Herstellungsvorgangs ausgeführten Kommandos.

`-clean`

veranlaßt das Löschen des für die Herstellung angelegten Verzeichnisbaums nach Abschluß des Build-Prozesses.

`-sign`

fügt eine PGP-Signatur zum RPM-Paket hinzu.

`-buildroot` *Verzeichnis*

veranlaßt die Erstinstallation der aufgebauten Software in dem angegebenen Verzeichnis. Die Build-Scripts müssen speziell für diese Option vorbereitet sein, damit der Herstellungsprozeß korrekt ablaufen kann.

`-timecheck` *Sekunden*

verhindert das Einpacken von älteren Dateien in das binäre RPM-Paket. Durch diese Option kann sichergestellt werden, daß nur die im Herstellungsprozeß neu erzeugten Dateien in das Binärpaket aufgenommen werden.

## Siehe auch:

`glint(8)`, `rpm2cpio`, `dpkg(8)` und `dselect(8)`

---

[Next](#) [Up](#) [Previous](#) [Contents](#) [Index](#)

**Next:** [shutdown](#) **Up:** [Die Kommandos für root](#) **Previous:** [rmmod](#)



Next: [umount](#) Up: [Die Kommandos für root](#) Previous: [rpm](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# shutdown

## Funktion:

**shutdown** fährt das System herunter

## Syntax:

**shutdown** [-h/-r] [-fqs] [now/hh:ss/+Minuten]

## Beschreibung:

**shutdown** fährt das System herunter und sorgt so für ein geregeltes Ende des Betriebes.

Alle Prozesse erhalten zuerst ein SIGTERM als Warnung. Viele Programme fangen dieses Signal ab und schließen ihre Aktivität ordentlich ab bevor sie terminieren. Schließlich werden alle

Benutzerprozesse mit SIGKILL beendet, das Dateisystem demontiert  und das System rebootet oder angehalten.

Wenn keine Zeit angegeben ist, wird das System nach zwei Minuten heruntergefahren. Das shutdown Kommando legt automatisch die Datei /etc/nologin an, die vom login ausgewertet wird und das Einloggen normaler Anwender während des shutdown verhindert.

Es gibt eine Reihe von Links auf shutdown, die jeweils verschiedene Optionen voreingestellt haben. Alle Links fahren das System sofort herunter, wenn keine Zeitspanne auf der Kommandozeile angegeben wird.

halt

hat die Option ``-h'` gesetzt

reboot

hat die Option ``-r'` gesetzt

fasthalt

hat die Optionen ``-h'` und ``-f'` gesetzt

fastboot

hat die Optionen `-r` und `-f` gesetzt

Neben dem hier beschriebenen shutdown-Programm von Peter Orbaek gibt es noch ein shutdown-Kommando von Miquel van Smoorenburg, das als Front-End das separate Programm `halt` oder `reboot` vom gleichen Autor aufruft. Eine Besonderheit des shutdown-Front-Ends ist der Aufruf des Programms (Shellscripts) `brc` zum „Aufräumen“ des Systems und zum Abbauen des Dateisystems.

## Optionen:

`-h`

hält das System an, ohne neu zu booten

`-r`

startet das System nach dem Herunterfahren sofort neu

`-f`

erzeugt die Datei `/etc/fastboot`, die in der Systeminitialisierungsdatei gesucht werden kann, um gegebenenfalls die Prüfung der Dateisysteme zu überspringen; diese Datei wird nicht automatisch gelöscht

`-q`

unterdrückt die Frage nach einer Begründung für die Unterbrechung des Rechnerbetriebes (Meldung für `wall`)

`-s`

erzeugt die Datei `/etc/singleboot`, die vom `simpleinit` ausgewertet wird und zum Systemstart im Einbenutzermodus führt

`now`

fährt das System sofort herunter

***hh:mm***

fährt das System zu der angegebenen Uhrzeit herunter

***+Minuten***

fährt das System nach Ablauf der Minuten herunter

## Autor:

Peter Orbaek

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [umount](#) **Up:** [Die Kommandos für root](#) **Previous:** [rpm](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

**Next:** [Systemverwaltung](#) **Up:** [Die Kommandos für root](#) **Previous:** [shutdown](#)

## Subsections

- [Funktion:](#)
  - [Syntax:](#)
  - [Beschreibung:](#)
  - [Optionen:](#)
- 

# umount

## Funktion:

**umount** setzt ein aufgesetztes Dateisystem ab

## Syntax:

**umount** *[-a] [-t minix | ext| msdos]*

**umount** *Gerätedatei | Verzeichnis*

## Beschreibung:

**umount** setzt das aufgesetzte Dateisystem des mit der *Gerätedatei* verbundenen Gerätes von dem *Verzeichnis* ab.

Beispielsweise muß eine Diskette mit Dateisystem erst auf diese Weise abgesetzt werden, bevor sie aus dem Laufwerk genommen werden kann. **umount** führt automatisch einen `sync` Befehl zur Sicherung aller im Cache gehaltenen Daten aus. Weil das eine Weile dauern kann, ist es von großer Wichtigkeit, mit dem Herausnehmen einer Diskette so lange zu warten, bis der Schreibvorgang beendet ist.

**umount** kann nur inaktive Dateisysteme absetzen. Das bedeutet, daß kein anderes Dateisystem auf dem abzusetzenden System aufgesetzt sein darf, und daß kein Prozeß ein Verzeichnis des abzusetzenden Systems als Arbeitsverzeichnis benutzen darf. Insbesondere darf sich kein Benutzer in dem abzusetzenden Dateisystem aufhalten. Andernfalls wird eine Meldung der Form `...device busy` ausgegeben.

## Optionen:

**-a**

setzt alle in `/etc/fstab` aufgeführten Devices ab (auch das Root-Filesystem)

**-t *Typ***



setzt nur Dateisysteme vom *Typ* ab. Beschreibung des Parameters *Typ* ist beim Kommando `mount` zu finden.

---

[Next](#) [Up](#) [Previous](#) [Contents](#) [Index](#)

**Next:** [Systemverwaltung](#) **Up:** [Die Kommandos für root](#) **Previous:** [shutdown](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

# Systemverwaltung

Bei der Installation einer Linux-Distribution wird von den halbautomatischen Installationsprogrammen im Allgemeinen ein gebrauchsfertiges System hergestellt. Sogar die Netzwerkkonfiguration ist meistens so weit komplett, daß sofort eine Verbindung zu einem auf Ethernet basierenden lokalen Netz aufgenommen werden kann.

Trotzdem ist eine solche Installation in der Regel nicht fertig. Eventuell muß zusätzliche Software installiert werden, weitere Accounts müssen eingerichtet, neue Hardware muß angeschlossen und in das System integriert werden. In vielen Fällen sind Anpassungen an die individuellen Wünsche der Benutzer erforderlich. Es müssen gelegentlich Fehler behoben und ``Wartungsarbeiten" durchgeführt werden.

Diese und weitere Aufgaben, die notwendig sind, um die Produktivität des Linux-Systems herzustellen, zu verbessern und zu erhalten, werden allgemein unter dem Begriff Systemverwaltung zusammengefaßt. Bei den meisten Linux-Systemen wird es keine besondere Person geben, die sich auf die Systemverwaltung spezialisiert. ``Die Systemverwalterin" ist deshalb eine funktionale Rolle, die üblicherweise von Personen übernommen wird, die zusätzlich als ganz normale User mit dem System arbeiten.

## Der privilegierte `root`-Account

Die Systemverwalterin braucht für viele ihrer Aufgaben besondere Rechte. Dazu steht ihr ein besonderer Account mit dem Benutzernamen `root` zur Verfügung. Als `root` ist die Systemverwalterin ``Superuser". Sie hat dann Privilegien, die es ihr erlauben, uneingeschränkt auf alle Daten, Geräte und Komponenten im System zuzugreifen. So kann sie Veränderungen am System vornehmen, die allen normalen Benutzern von den Sicherheitsmechanismen des Linux-Kernels verboten werden.

Um Superuser zu werden, gibt es zwei Wege:

1.

Sie können sich beim Login als User `root` anmelden und erhalten nach Eingabe des korrekten Rootpaßwortes eine Shell mit Rootrechten.

2.

Sie können mit dem Programm [su](#) aus einer normalen Shell heraus eine neue Shell mit Rootrechten bekommen. `su` fragt wie das `login`-Programm nach einem Paßwort und gibt die Rootshell erst, nachdem das korrekte Rootpaßwort eingegeben wurde.

Der Superuser hat mehr Rechte als die normalen Systembenutzer. Das bedeutet aber absolut nicht, daß der `root`-Account *besser* ist. Weil all die sinnvollen und hilfreichen Schutz- und Sicherheitsmechanismen des Betriebssystems für den Superuser nicht zur Verfügung stehen, ist das Arbeiten mit Rootrechten sehr riskant. Sie sollten deshalb nur die Befehle als `root` ausführen, die wirklich die Privilegien dieses Accounts benötigen.

- 
- [Der privilegierte `root`-Account](#)
- 

[Next](#) [Up](#) [Previous](#) [Contents](#) [Index](#)

**Next:** [Der Anfang und](#) **Up:** [Das Linux Anwenderhandbuch](#) **Previous:** [umount](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

## Subsections

- [Von Diskette booten](#)
  - [Von Festplatte booten](#)
  - [LILO](#)
    - [Installation](#)
    - [Bootkonzepte 1: wohin mit dem Bootloader?](#)
    - [Bootkonzepte 2: woher mit dem Betriebssystem?](#)
    - [Eine Beispielkonfiguration](#)
    - [Konfigurationsdatei](#)
    - [LILO deinstallieren](#)
  - [Der Bootprompt](#)
    - [Allgemeine Einstellungen](#)
    - [RAM-Disk](#)
    - [Festplatten und IDE-Controller](#)
    - [SCSI-Kontroller](#)
    - [SCSI-Bandlaufwerke](#)
    - [CD-ROM-Laufwerke](#)
    - [Diskettenlaufwerke](#)
    - [Verschiedenes](#)
    - [Busmäuse](#)
    - [Netzwerkkarten](#)
  - [Die Vorgänge bei der Kernelinitialisierung](#)
  - [RAM-Disk und Initrd](#)
  - [init](#)
    - [Die Konfigurationsdatei `/etc/inittab`](#)
    - [Respawn mit Initscript](#)
    - [Die Shellscrippte zur Systeminitialisierung](#)
    - [Auswahl und Änderung des Runlevel](#)
  - [Das System herunterfahren](#)
-

# Der Anfang und das Ende

Die erste umfangreiche Aufgabe der Systemverwaltung besteht darin, das System zu starten und in einen Anfangszustand zu bringen, der es auf die eigentlichen Anwendungen optimal vorbereitet. Der Vorgang des Systemstarts läßt sich in vier Phasen gliedern:

1.

Zuerst muß sich der Kernel in den Arbeitsspeicher laden. Dieser Vorgang wird als ``booten'' bezeichnet.

2.

Dann startet das Betriebssystem, nimmt den Prozessor, den Arbeitsspeicher und das Motherboard mit seinen Chips in Besitz und bereitet die geräteunabhängigen Datenstrukturen auf ihre Aufgaben vor.

3.


Danach erzeugt Linux seinen ersten Prozeß, der zunächst im Speicherbereich des Kernels bleibt und dort die in den Kernel eingebauten Gerätetreiber initialisiert, das Laufzeitsystem zum Laden von Programmen vorbereitet und die Dateisysteme aktiviert.

4.

Schließlich verläßt der erste Prozeß den Speicherbereich des Kernels und führt das erste Programm im ``Userspace'' aus. Dieses Programm ist dafür verantwortlich, die Services und Arbeitsumgebungen des Laufzeitsystems einzurichten.

Diese vier aufeinander aufbauenden Phasen bestimmen, wie das Laufzeitsystem aussieht. Als Systemverwalterin müssen Sie dafür sorgen, daß alle Voraussetzungen für den produktiven Betrieb des Linux-Systems erfüllt sind.

Im Moment des Einschaltens ist der Arbeitsspeicher des Rechners vollständig leer. Das Betriebssystem ist nicht in irgendwelchen Chips fest auf der Hauptplatine gespeichert, sondern es befindet sich zum Beispiel auf einer Diskette und muß erst von dort geladen werden.

Von wo und auf welche Weise das Betriebssystem geladen wird, ist von der Rechnerarchitektur abhängig. Linux für Intel-PC wird normalerweise von Diskette oder Festplatte gebootet. Manchmal wird der PC auch als ``*Diskless Workstation*'' eingesetzt, die das Betriebssystem über eine Netzwerkverbindung bootet  und das Rootfilessystem per NFS importiert.

Die Intel-PC sind mit einem *Basic Input Output System* (BIOS) ausgestattet, das beim Einschalten oder nach einem Reset automatisch versucht, den Bootvorgang zu starten. Das BIOS überläßt die Einzelheiten des Ladevorgangs dem Betriebssystem selbst, stellt aber einfache Funktionen zum Lesen von Diskette oder Festplatte und zum Schreiben in den Speicher oder auf den Bildschirm zur Verfügung.

Das Betriebssystem wird nicht gleich als ganzes in den Speicher geladen. Stattdessen wird vom BIOS ein sehr kleines Programm, ein sogenannter ``*Bootloader*'', in den Arbeitsspeicher geladen und die Kontrolle für den weiteren Bootvorgang an dieses Programm abgegeben.

# Von Diskette booten

Im einfachsten Fall lädt das BIOS einen Linux-spezifischen Bootloader vom ersten Sektor einer Diskette im Bootlaufwerk. Der Loader zieht den Kernel hinterher, schiebt ihn an die richtige Stelle im Arbeitsspeicher und übergibt schließlich die Kontrolle über den Rechner an das Betriebssystem.

Der erste Block einer Kerneldatei enthält sinnvollerweise genau das passende Programm zum Laden des Kernels. Wenn der Kernel mit `dd` oder `cp` auf eine formatierte, rohe Diskette ohne Dateisystem geschrieben wird, befindet sich der Bootblock genau an der richtigen Stelle, um vom BIOS gelesen zu werden. Sie erhalten also eine einfache Linux-Bootdiskette, indem Sie eine Kerneldatei auf eine Diskette kopieren.

Bei dieser Methode haben Sie keine Möglichkeit, dem Kernel zusätzliche Bootparameter zu übergeben. Bis auf die wenigen Parameter, die sich mit dem Programm [rdev](#) an der Kerneldatei verändern lassen, müssen alle einkompilierten Kerneinstellungen zu Ihrem System passen, damit diese Bootmethode erfolgreich ist.

Wenn Sie Ihr Linux-System selbst installiert haben, ist Ihnen bereits eine andere Methode zum Booten von Diskette begegnet. Bei den Installationsdisketten für die Linux-Distributionen wird meistens der LILO Bootloader verwendet, der später noch genauer beschrieben wird.

Diese Methode, von einer Diskette zu booten, hat zwei Vorteile:

- Der LILO Bootloader ermöglicht die Eingabe einer Kommandozeile für den Kernel auf dem ``Bootprompt''. Damit können Einstellungen an Gerätetreibern und andere Anpassungen des Kernels vorgenommen werden, ohne die das System möglicherweise nicht korrekt hochfährt.
- Eine mit LILO gebootete Diskette kann ein Dateisystem mit beliebigen Programmen und Daten enthalten. So ein arbeitsfähiges Minimalsystem auf Diskette, wie Sie es bei der Installation kennengelernt haben, kann eventuell als ``Rettungsinsel'' dienen, wenn das System auf der Festplatte aus irgendeinem Grund nicht mehr funktionsfähig ist.

Wenn Sie sich solch eine Rettungsinsel herstellen wollen, können Sie mit einem der vorgefertigten Images beginnen, die bei einigen Linux-Distributionen enthalten sind. Suchen Sie in dem Verzeichnis auf der Distributions-CD oder auf dem FTP-Server, wo Sie die Daten für die Root-Disk zur Installation gefunden haben, nach einer Datei mit dem Namen `rescue`. Stellen Sie aus dieser Datei eine Root-Disk her, so wie Sie es für die Installationsdisketten getan haben. Zusammen mit der originalen Bootdiskette von der Installation haben Sie dann ein arbeitsfähiges Minimalsystem auf Diskette.

Wenn Sie eigene Bootdisketten für spezielle Aufgaben herstellen möchten, finden Sie weitere Informationen und Anregungen im `Bootdisk-HOWTO`.

# Von Festplatte booten

Normalerweise wird Linux dauerhaft auf der Festplatte installiert und soll auch von dort gebootet werden. Das bringt einen kleinen Zeitgewinn beim Lesen der Kerneldatei und es vermindert die Anzahl der auf dem Schreibtisch herumliegenden Disketten.

Linux ist häufig nicht das erste oder einzige Betriebssystem für einen PC. Deshalb wurden für Linux

mehrere unterschiedliche Methoden und Programme entwickelt, mit denen die Anpassung des Bootkonzepts an die Vielfalt unterschiedlicher Konfigurationen und Einsatzgebiete für Intel-PC möglich ist. Welcher Weg der Richtige für Sie ist, hängt vor allem davon ab, wie Sie Ihre Festplatte eingeteilt haben.


Anders als Disketten werden Festplatten in Verwaltungsbereiche, sogenannte Partitionen unterteilt. Dieses Konzept wird von allen Betriebssystemen für PC unterstützt. Eine Festplatte kann in maximal vier primäre Partitionen eingeteilt werden, eine dieser Partitionen läßt sich als ``erweiterte Partition" zusätzlich in ``logische" Partitionen (Laufwerke) unterteilen. Jede Partition kann ein eigenes Dateisystem tragen und von einem anderen Betriebssystem besiedelt werden.

Zu Anfang unterscheidet sich der Vorgang beim Booten von Festplatte nicht wesentlich von dem beim Booten von Diskette: wenn das BIOS keine Bootdiskette im Laufwerk A: findet, lädt es automatisch den ersten Sektor der ersten Festplatte (den *Master Boot Record*, MBR) und führt den darin enthaltenen Bootloader aus. Der Bootloader im Master Boot Record einer Festplatte ist ebenso austauschbar wie der einer Bootdiskette, er wird zusammen mit dem ersten Betriebssystem installiert. Im Prinzip ist also auch hier der MBR ein Teil des Betriebssystems und das Betriebssystem damit selbst dafür verantwortlich, sich in den Arbeitsspeicher zu laden.

Offensichtlich birgt dieses Verfahren ein gewisses Konfliktpotential, wenn ein zweites oder drittes Betriebssystem auf der gleichen Festplatte installiert wird. Um die friedliche Koexistenz zu ermöglichen, wird ein zusätzlicher Zwischenschritt vor dem Laden des eigentlichen Betriebssystems eingeführt: beispielsweise holt der bei einer DOS- oder Windows-Installation im Master Boot Record installierte Bootloader aus dem ersten Sektor der als *aktiv* markierten (DOS-)Partition einen zweiten Bootloader in den Speicher und überläßt diesem *Secondary Bootloader* das Laden des Betriebssystems.

Für Linux stehen mehrere Bootkonzepte zur Auswahl, mit denen Sie sehr flexibel auf die Gegebenheiten und Anforderungen eines geteilten Systems reagieren können.


- Sie können einen linuxspezifischen Bootloader im Bootsektor einer primären Linuxpartition installieren. Dieser sekundäre Bootloader kann entweder vom primären Bootloader im MBR oder vom Bootmanager eines anderen Betriebssystems aufgerufen werden, um Linux als Betriebssystem zu laden.
- Sie können einen Linux Bootloader im Master Boot Record installieren und diesen als Bootmanager verwenden, um Linux oder andere Betriebssysteme über deren Secondary Bootloader zu laden.
- Sie können Linux einfach von DOS aus starten und damit die Installation eines zusätzlichen Bootloaders vermeiden.

Als allgemeiner Bootloader für Linux, der sowohl als Bootmanager im Master Boot Record als auch als Secondary Bootloader eingesetzt werden kann, hat sich LILO von Werner Almesberger bewährt. 

Zum Booten von Linux aus einem laufenden DOS-System heraus gibt es das Programm `loadlin.exe`  von Hans Lermen.

## LILO

Mit dem *generischen Bootloader für Linux*, kurz Linux Loader oder LILO, können Sie Linux und eine

ganze Reihe anderer Betriebssysteme für Intel-PC  von Festplatte booten. Als ``Bootmanager" kann er bis zu 16 verschiedene Systeme oder Konfigurationen starten, er kann Linux von der zweiten oder dritten Festplatte laden, er übergibt dem Kernel eine Kommandozeile und er unterstützt die Init-Ramdisk.

Die Vielseitigkeit von LILO beruht vor allem darauf, daß er mit einer als *map* bezeichneten Tabelle alle Daten, die er zum Laden des Betriebssystems benötigt, von (fast) beliebigen Plattenblöcken holen kann. Vorausgesetzt wird lediglich, daß der Bootsektor von LILO auf einer primären Partition der ersten Festplatte installiert ist und daß sich die Daten in einem vom BIOS sichtbaren Teil der Festplatte befinden.

## Installation


Der Linux Loader gehört zur Basisausstattung eines Linux-Systems, das bei allen Linux-Distributionen automatisch auf die Festplatte kopiert wird. Die eigentliche Installation von LILO geschieht durch den Aufruf des *Map-Installers* mit dem Kommandonamen `lilo`. `lilo` liest die Konfigurationsdatei `/etc/lilo.conf` und schreibt den darin enthaltenen Einstellungen entsprechend den Bootloader in einen Bootsektor auf der Festplatte.

Die Installationsprogramme der Linux-Distributionen versuchen mit mehr oder weniger hilfreicher Benutzerführung, die Konfigurationsdatei für LILO zu erstellen und damit das Linux-System von Festplatte bootfähig zu machen. Wenn Sie LILO auf diese Weise installieren und dabei auf Probleme oder Unklarheiten stoßen, kann Ihnen dieses Kapitel in den meisten Fällen weiterhelfen. Wenn Sie unsicher sind, sollten Sie auf die Installation von LILO auf der Festplatte ``im ersten Anlauf" verzichten und sich in Ruhe die gesamte Dokumentation zu LILO durchlesen, bevor Sie die Installation zu einem passenden Zeitpunkt durchführen.

Bei allen bekannten Linux-Distributionen finden Sie wenigstens einen Teil der Hilfstexte zu LILO in einem Unterverzeichnis von `/usr/doc`. Die vollständige Dokumentation finden Sie jedenfalls bei den Sourcen, die bei den auf CD vertriebenen Linux-Distributionen mitgeliefert werden. Wenn Sie Zugang zum Internet haben, finden Sie eine Online-Version auf <http://www.lunetix.de>.

## Bootkonzepte 1: wohin mit dem Bootloader?

LILO kann als primärer oder als sekundärer Bootloader auf der Festplatte oder im Bootsektor einer Diskette installiert werden. Als primärer Loader wird er in den Master Boot Record der ersten Festplatte geschrieben und beim Systemstart automatisch vom BIOS gestartet. Als sekundärer Loader wird er im Bootsektor einer primären Partition der ersten Festplatte installiert und muß vom primären Bootloader in den Speicher geholt und gestartet werden. Im Bootsektor einer Diskette installiert, wird LILO gestartet, wenn diese Diskette beim Einschalten des Rechners im Bootlaufwerk liegt.

Wenn er im Master Boot Record installiert wird, überschreibt LILO jeden dort eventuell von einem anderen Betriebssystem installierten Primary Bootloader. LILO kann in der Regel  die Funktion des verdrängten Bootloaders mit übernehmen, wenn die entsprechenden Einstellungen in der Konfigurationsdatei `/etc/lilo.conf` vorgenommen wurden.

Als sekundärer Bootloader kann LILO nur auf der ersten Festplatte und dort nur auf einer primären Linux-Partition mit einem beliebigen Linux-Dateisystem oder im Bootsektor der erweiterten Partition installiert werden. LILO kann nicht auf einer Swap-Partition oder im Bootsektor einer von einem anderen Betriebssystem belegten Partition installiert werden.





Damit LILO als sekundärer Bootloader arbeiten kann, muß im Master Boot Record ein primärer Bootloader enthalten sein, der den Linux Loader als sekundären Bootloader ausführt. Zu diesem Zweck eignet sich beispielsweise der mit DOS installierte Primay Bootloader. Damit der DOS-Loader den LILO ausführt, muß die Partition, auf der LILO installiert wurde, die einzige aktive Partition der ersten Festplatte sein. Die primären Partitionen können mit dem Programm `fdisk` (sowohl unter DOS als auch unter Linux) aktiviert werden. Die erweiterte Partition läßt sich nur mit dem `fdisk` von Linux aktivieren.

Unabhängig davon, ob LILO als primärer oder sekundärer Bootloader auf der ersten Festplatte oder im Bootsektor einer Diskette installiert wurde, kann der Linux Loader als Bootmanager verschiedene Linux-Systeme und andere Betriebssysteme booten. Beispielsweise kann der Linux Loader DOS booten, indem er den sekundären Bootloader von der DOS-Partition ausführt.

## Bootkonzepte 2: woher mit dem Betriebssystem?

Die Lokalisierung eines geeigneten Platzes für den Linux-Kernel selbst ist weitgehend unabhängig von dem Ort, an dem sich der Linux Loader oder die Partition mit dem Linux-Dateisystem befindet.

Der Linux Loader lädt den Kernel von der Festplatte, indem er die durch ihre Zylinder/Kopf/Sektor-Koordinaten identifizierten Datenblöcke einzeln einliest und in der richtigen Reihenfolge in den Arbeitsspeicher kopiert. Dabei benötigt der Loader keine Information über Partitionen, Dateisysteme, Verzeichnisse oder Dateinamen. 

Diese Methode ermöglicht maximale Flexibilität bei der Unterbringung der Kerneldatei: LILO kann Linux von jeder Stelle des Systems laden, die er lesen kann. Leider hat LILO zum Zeitpunkt seiner Arbeit nicht die Gerätetreiber und Dienste von Linux zur Verfügung. Stattdessen muß der Loader mit den Funktionen auskommen, die ihm vom BIOS angeboten werden. Deshalb müssen sich der Kernel und alle sonstigen für das Booten notwendigen Daten  in dem für das BIOS sichtbaren Systembereich befinden.

Wie dieser Bereich konkret aussieht, hängt von der verwendeten Hardware ab. In jedem Fall erkennt das BIOS die beiden im CMOS eingetragenen IDE-Festplatten, von denen es aber nur jeweils die ersten 1024 Zylinder lesen kann. Dieser ``Sehfehler" führt zu der auch bei DOS zu beobachtenden Einschränkung des BIOS-Horizonts auf 504 MB Festplattenplatz.

Die BIOS-Versionen neuerer Motherboards sind manchmal in der Lage, vier IDE-Platten zu erkennen, und sie akzeptieren größere Zylinder. Auf diese Weise können sich zusätzliche Möglichkeiten für die Unterbringung des Linux-Kernels ergeben. Weitere Informationen zu großen IDE-Festplatten finden Sie bei der Kernel-Dokumentation in der Datei `/usr/src/linux/Documentation/ide.txt`.

SCSI-Hostadapter installieren beim Einschalten des Rechners in der Regel eine BIOS-Erweiterung, die das Booten von SCSI-Festplatten ermöglicht. Ob mehr als zwei dieser Festplatten von der BIOS-Erweiterung angesprochen werden können und wie groß der vom BIOS lesbare Plattenbereich ist, hängt vom Hostadapter ab.

Wenn Sie innerhalb des BIOS-Horizonts keine Linux-Partition anlegen können, bleibt Ihnen noch die Möglichkeit, die Kerneldatei und die zum Booten notwendigen Dateien auf der primären DOS-Partition (Laufwerk C:\) unterzubringen. Dazu müssen Sie die DOS-Partition in das Dateisystem Ihres laufenden Linux-Systems einbinden, das Verzeichnis `/boot` samt Inhalt an eine geeignete Stelle auf der DOS-Partition verschieben, einen symbolischen Link von `/boot` zu dem neuen Verzeichnis anlegen und natürlich noch den Kernel auf die DOS-Partition kopieren. Vor dem Aufruf des

Map-Installers müssen Sie noch den Eintrag für die Kerneldatei entsprechend ändern.

Bitte beachten Sie, daß die Linux-Dateien beim Defragmentieren der DOS-Partition verschoben werden können und danach das Mapping nicht mehr stimmt. Durch Setzen des ``System"-Attributes unter DOS können Sie die Dateien vor dem Verschieben schützen.

## Eine Beispielkonfiguration

Wenn Sie die Konfiguration und Installation von LILO nicht der halbautomatischen Installationsprozedur Ihrer Linux-Distribution überlassen können/wollen, müssen Sie die Konfigurationsdatei `/etc/lilo.conf` ``von Hand" erzeugen und anschließend den Map-Installer `lilo` aufrufen.

Das folgende Beispiel zeigt eine solche Konfigurationsdatei für ein System, bei dem auf der ersten Partition einer IDE-Festplatte (`/dev/hda1`) MS-DOS und auf der zweiten (primären) Partition (`/dev/hda2`) Linux installiert ist.

```
boot=/dev/hda1
install=/boot/boot.b
map=/boot/map
vga=normal
prompt
timeout=50
image=/vmlinuz
    label=linux
    root=/dev/hda2
    read-only
    append=" "
other=/dev/hda1
    label=msdoof
    table=/dev/hda
```

In der Regel sollte es ausreichen, die Bezeichnungen der Partitionen (`/dev/hda2`, `/dev/hda1`) sowie den Namen des Kernel-Images dem konkreten System anzupassen.


In dem Beispiel wird LILO als sekundärer Bootloader auf die zweite Partition geschrieben. Diese Partition enthält das Root-Filesystem und ist deshalb zur Aufnahme des Bootloaders geeignet. Voraussetzung für die einwandfreie Funktion dieser Konfiguration ist, daß sich die Partition `/dev/hda2` vollständig in dem vom BIOS sichtbaren Bereich bis zum Zylinder Nummer 1024 befindet, daß der Master Boot Record einen primären DOS-Loader enthält und daß die Partition `/dev/hda2` die einzige aktive Partition ist.

Eine minimale Änderung der Konfigurationsdatei verwandelt das darin festgelegte Bootkonzept grundlegend: wenn die erste Zeile von `boot=/dev/hda2` in `boot=/dev/hda` verändert wird, wird LILO als primärer Bootloader im Master Boot Record installiert und ersetzt dort den primären DOS-Loader. In dem Beispiel hat diese Änderung keine praktische Konsequenz, trotzdem sollten Sie sich den Unterschied klar machen, damit Sie nicht aus Versehen etwas überschreiben, was Sie eigentlich lieber erhalten wollten.

Um den neuen Bootsektor zu schreiben, müssen Sie den Map-Installer `lilo` aufrufen:

```
# /sbin/lilo
Added linux *
Added msdoof
# _
```

Auf diese Weise muß `lilo` immer aufgerufen werden, wenn ein neues Kernel-Image installiert werden soll oder wenn eine der Dateien aus dem Verzeichnis `/boot` verändert worden ist. Allein das Überschreiben der alten Kerneldatei durch die neue reicht nicht aus, weil damit nicht sichergestellt ist, daß die neue Kerneldatei exakt die gleichen Datenblöcke auf der Festplatte belegt.

Wenn Sie LILO im Master Boot Record installieren und dabei den primären Bootloader eines anderen Betriebssystems überschreiben, möchten Sie möglicherweise eine Sicherheitskopie des alten Bootsektors behalten. LILO erzeugt automatisch eine solche Kopie unter dem Namen  `/boot/boot.0300` und kann damit den alten Zustand wieder herstellen.

Sie können sich auch eine Sicherungskopie auf Diskette ziehen, mit der Sie einen indirekten Festplattenstart mit dem alten Bootloader durchführen können. Wie Sie so eine ``Bootdiskette'' erzeugen, zeigt das folgende Beispiel:

```
# dd if=/dev/hda of=/dev/fd0 bs=512 count=1
1+0 records in
1+0 records out
# _
```

## Konfigurationsdatei

Das oben gezeigte Beispiel einer Konfigurationsdatei kann meistens mit leichten Veränderungen übernommen werden. Die folgende Liste erklärt Ihnen die Funktion der einzelnen Einträge. Der Linux Bootloader erlaubt noch wesentlich mehr Einstellungen. Eine vollständige Liste finden Sie im Dokument von W. Almesberger.

### **boot=***Partition*

Diese Variable bestimmt die Partition, auf die der Bootloader geschrieben werden soll. Wenn diese Angabe in der Konfigurationsdatei fehlt, wird der Bootsektor auf die aktuelle Rootpartition geschrieben.

### **install=***Bootsektor*

Mit dieser Variablen kann die Datei ausgewählt werden, die als neuer *Bootsektor* geschrieben werden soll. Falls diese Angabe fehlt, wird der Bootsektor aus `/dev/boot.b` gelesen.

### **delay=***Zehntelsekunden*

Mit der *delay*-Variablen kann die Zeit eingestellt werden, die LILO auf eine der Tasten zur Ausgabe des Bootprompts wartet. Wenn diese Zeit verstrichen ist und die Konfigurationsdatei das Schlüsselwort `prompt` nicht enthält, wird automatisch das erste in der Konfigurationsdatei definierte Kernelimage geladen.

### **prompt**

wenn dieses Schlüsselwort in der Konfigurationsdatei auftaucht, gibt LILO bei jedem Systemstart den Bootprompt aus und wartet die in `timeout` festgelegte Zeitspanne auf die Eingabe einer Kommandozeile für den Kernel. Fehlt dieses Schlüsselwort, können Sie den Bootprompt erhalten, indem Sie in der durch `delay` festgelegten Zeit nach dem Start von LILO eine der Tasten ALT, CONTROL oder SHIFT drücken oder die Schalter CAPS-LOCK/SCROLL-LOCK

betätigen.

### **timeout=Zehntelsekunden**

Mit dieser Variablen wird die Zeit eingestellt, die LILO nach dem Bootprompt auf eine Eingabe wartet, bevor er zum automatischen Laden der ersten konfigurierten Kerneldatei zurückkehrt. Wenn die Variable `timeout` nicht gesetzt ist, wartet LILO endlos auf eine Eingabe.

### **image=Kerneldatei**

Mit dem Eintrag `image` wird eine *Kerneldatei* (mit absolutem Pfad) angegeben. Dieser Eintrag leitet einen speziellen Teil in der Konfigurationsdatei ein. Alle Einträge bis zum nächsten `image` oder bis zum Dateiende gelten nur für diese Kerneldatei. Die folgenden kernelspezifischen Einstellungen können vorgenommen werden:

#### **label=Name**

Zur Auswahl einer von mehreren konfigurierten Kerneldateien wird nach dem Bootprompt (siehe unten) entweder die Kerneldatei oder der unter `label` angegebene *Name* angegeben. Mit Hilfe vom `label` kann ein `image` auch mehr als einmal benutzt werden.

#### **vga={normal | extended | ask}**

Der Kernel wird in dem angegebenen Textmodus gestartet. `normal` ist der Textbildschirm mit 80x25 Zeichen, `extended` liefert 80x50 Zeichen und `ask` veranlaßt den Kernel, ein Menü mit allen möglichen Videomodi anzuzeigen und auf eine Auswahl zu warten.

Wenn der Eintrag für den VGA-Modus in der Konfigurationsdatei weggelassen wird, startet der Kernel mit dem in der Kerneldatei festgelegten Videomodus. Diese Einstellung läßt sich mit dem Systemprogramm [rdev-Kommando](#) verändern.

#### **ramdisk=Kilobytes**

Dieser Eintrag wird an den Kernel weitergegeben, der damit die maximale Größe der RAM-Disks von 2048kB auf den angegebenen Wert ändert. Da die RAM-Disks allein durch die Initialisierung beim Booten noch keinen Speicher belegen, kann der vom Kernel vorgegebene Wert ohne Nachteile belassen werden.

Da der Treiber für die RAM-Disk nicht obligatorischer Bestandteil des Kernels ist, geht dieser Eintrag ins Leere, wenn der Treiber nicht vorhanden ist.

#### **root=Rootfilesystem**

Die Variable `root` enthält den Namen der Gerätedatei für die Festplattenpartition mit dem Rootfilesystem. Wenn anstelle einer Gerätedatei das Wort `current` angegeben ist, wird die aktuelle Rootpartition angenommen. Wenn dieser Eintrag ganz fehlt, wird die beim Übersetzen des Kernels bestimmte Rootpartition beibehalten.

#### **append=Zeichenkette**

Mit `append` wird eine Zeichenkette festgelegt, die bei jedem Start dieses Kernel-Images an die Kommandozeile angehängt wird.

#### **read-only**

Das Schlüsselwort `read-only` sorgt dafür, daß der Kernel das Rootfilesystem im ersten Anlauf nur zum Lesen mountet. Dadurch hat das Systemprogramm zum Checken des Dateisystems die Möglichkeit, Fehler zu reparieren, ohne durch Schreiboperationen des Kernels gestört zu werden.

Wenn die speziellen Parameter für alle Konfigurationen gleich sind, können die entsprechenden Einträge auch im allgemeinen Teil vorgenommen werden.

Sie können mit LILO auch ein anderes Betriebssystem, beispielsweise MS-DOS, booten. Dazu dienen die folgenden zusätzlichen Variablen:

#### ***other=Device***

Die Variable `other` leitet einen speziellen Teil der Konfigurationsdatei ein, wie `image`. Die hier angegebene Partition muß den Secondary Bootloader für das andere Betriebssystem enthalten. Beispielsweise muß bei MS-DOS hier die primäre DOS-Partition angegeben werden.

#### ***loader=chain\_loader***

Innerhalb des von `other` eingeleiteten speziellen Teils wird mit der Variablen `loader` ein spezielles Programm bestimmt, das besondere Vorbereitungen zum Umschalten der Betriebssysteme treffen kann. Das Standardprogramm (Voreinstellung) `chain.b` übergibt einfach die Kontrolle an den Secondary Bootloader des fremden Betriebssystems.

#### ***table=Device***

In der Variablen `table` kann die Festplatte angegeben werden, aus der die Partitionstabelle für das fremde Betriebssystem gelesen werden kann. Im Fall von MS-DOS kann diese Angabe unterbleiben, weil eine Art interner Partitionstabelle im DOS-Dateisystem gespeichert ist.

## **LILO deinstallieren**

Bei einem Programm wie dem Linux Bootloader, das sich buchstäblich am Rand des Geltungsbereichs der Betriebssysteme aufhält, taucht gelegentlich die Frage nach der Entfernung (Deinstallation) auf. Wenn Sie LILO im Master Boot Record installiert haben, kann er durch Neupartitionieren der Festplatte nicht entfernt werden.

Normalerweise legt `lilo` vor dem Überschreiben des Bootsektors eine Sicherheitskopie unter dem Namen `/boot/boot.HHUU` an, wobei `HH` die Hauptgerätenummer, `UU` die Untergerätenummer der Festplattenpartition ist, deren Bootsektor überschrieben wird. Wenn die Linux-Partition noch funktionsfähig ist, können Sie den Map-Installer `lilo` auch zum Entfernen des Bootloaders benutzen, indem Sie ihn einfach mit der Option ``uninstall' (-u)` aufrufen und damit den gesicherten alten Bootsektor zurückschreiben. Wenn Sie beispielsweise LILO vom Master Boot Record der ersten IDE-Platte löschen wollen, sieht das Kommando so aus:

```
# /sbin/lilo -s /boot/boot.0300 -u /dev/hda
LILO version 0.16, Copyright 1992-1995 Werner Almesberger
```

```
Reading boot sector from /dev/hda
Reading old boot sector.
Restoring old boot sector.
# _
```

Die Angabe von `-s /boot/boot.0300` kann entfallen, wenn die Sicherheitskopie wie in dem Beispiel unter dem von `lilo` selbst erzeugten Namen abgelegt ist.

`lilo` restauriert den Bootsektor nicht, wenn die Sicherheitskopie älter ist als der aktuelle Bootloader. Das ist zum Beispiel der Fall, wenn `lilo` wiederholt aufgerufen wird, um veraltete Kernelversionen durch aktuelle zu ersetzen. Bei diesen Updates wird eine existierende Sicherheitskopie nicht überschrieben. Das ist auch sinnvoll, denn mit der Kopie eines alten Linux Bootloaders können Sie den

ursprünglichen" Zustand des Bootsektors nicht wiederherstellen. Wenn Sie die Partitionierung der Festplatte seit der ersten Installation von LILO nicht verändert haben, sollte es keine Probleme mit der Verwendung der ältesten Sicherheitskopie geben. Sie können dann mit der Option `-U` anstelle von `-u` die Überprüfung der Zeitmarke verhindern.

Wenn Sie eine Sicherheitskopie des Bootsektors auf Diskette angelegt haben, können Sie einfach diese Sicherheitskopie wieder zurückschreiben. Die Umkehrung des in dem früheren Beispiel gezeigten Sicherungsprozesses sieht folgendermaßen aus:

```
# dd if=/dev/fd0 of=/dev/hda bs=512 count=1
1+0 records in
1+0 records out
# _
```

Vergewissern Sie sich, daß der auf Diskette gespeicherte Bootsektor tatsächlich noch funktioniert, indem Sie das System mit dieser Bootdiskette starten.

Wenn Sie keine brauchbare Sicherheitskopie des MBR haben, können Sie den Primary Bootloader vom MS-DOS ab Version 5.0 mit dem folgenden (DOS-)Kommando restaurieren:

```
A:\ >fdisk /mbr
A:\ >
```

Das Programm `mkboot` aus dem XIAFS-Paket bietet eine ähnliche Option.

## Der Bootprompt

Linux ist in der Lage, vom Bootloader eine Art Kommandozeile zu übernehmen und diese während des Kernelstarts auszuwerten. Mit den Argumenten auf dieser Kommandozeile können Gerätetreiber eingestellt und verschiedene Kerneloptionen geändert werden. Dieser Mechanismus zur Laufzeitkonfigurierung des Linux-Kernels ist vor allem bei den generischen Kernels auf den Bootdisketten der Linux-Distributionen höchst wertvoll, um einen Rechner mit einer problematischen Hardwarekonfiguration zum Laufen zu bringen.

Mit LILO erhalten Sie die Möglichkeit zur Eingabe einer solchen Kommandozeile, wenn in der LILO-Konfigurationsdatei das Schlüsselwort `prompt` eingetragen ist oder wenn Sie beim Booten eine der Tasten ALT, CONTROL oder SHIFT gedrückt halten.

Vor dem Laden des Kernels erscheint dann der typische LILO Bootprompt und signalisiert, daß der Loader auf die Kommandozeile wartet.

```
boot: _
```

Da LILO auch als Bootmanager arbeitet, erwartet er als erstes Argument auf der Kommandozeile das Label des Betriebssystems, das geladen werden soll. Eine Liste mit allen Labeln erhalten Sie, nachdem Sie die Tabulatortaste TAB gedrückt haben. In der oben besprochenen Beispielkonfiguration für LILO sind das `linux` und `msdoof`.

Im Anschluß an das Label können Sie beliebig viele Argumente für den Kernel übergeben, die Sie jeweils durch Leerzeichen voneinander trennen müssen. Diese Argumente werden unverändert an den Kernel weitergegeben, der seinerseits einige Parameter an die Programme des Laufzeitsystems übergeben kann.

Eine erste Gruppe von Argumenten überlagert die im Kernel-Makefile bzw. mit dem Systemprogramm `rdev` eingestellten Parameter.

`root=`**Root-Partition**

veranlaßt den Kernel, die angegebene Root-Partition zu mounten.

`read-only (ro)`

veranlaßt den Kernel, die Root-Partition Read-Only zu mounten.

`rw`

veranlaßt den Kernel, die Root-Partition mit Schreibberechtigung zu mounten, auch wenn in der Kerneldatei oder in der LILO-Konfigurationsdatei etwas anderes festgelegt ist.

`vga={normal | extended | ask}`

stellt den Videomodus für den Textbildschirm ein (die Modi entsprechen den beim Systemprogramm [rdev](#) beschriebenen).

Eine andere Gruppe von Argumenten dient der Konfiguration bestimmter Gerätetreiber. Wenn eines dieser Argumente auf der Kommandozeile auftaucht, wird die Setup-Funktion für den entsprechenden Gerätetreiber mit den hier angegebenen Parametern anstelle der in den Kernelsourcen festgelegten Vorgaben aufgerufen. Bei sehr vielen Treibern von Hardware-Controllern können die Werte für den Interrupt (*IRQ*) und die Speicheradresse für den Anfang des IO-Ports angegeben werden. Auf diese Weise lassen sich vom Standard abweichende Einstellungen konfigurieren und Hardwarekonflikte umgehen. Die Speicheradressen müssen immer als Hexadezimalzahlen mit dem Präfix `0x` angegeben werden.

Bei vielen Distributionen werden zur Installation des Betriebssystems generische Kernel benutzt, die in der Regel überflüssige Treiber enthalten. Wenn es beim Laden eines solchen Kernels zu Problemen mit einem Treiber für ein nicht vorhandenes Gerät kommt, können Sie diesen Treiber meistens abschalten, indem Sie als Argument für den entsprechenden Bootparameter einfach nur die Zahl 0 angeben.

## Allgemeine Einstellungen

`init=`**Programm**

erlaubt die Auswahl eines Programms, das anstelle von `/sbin/init` ausgeführt wird.

`reserve=`**Adresse,Größe**[**Adresse,Größe**...]

markiert bis zu fünf Adreßbereiche für IO-Ports als belegt und schützt sie so vor dem Zugriff beliebiger Gerätetreiber beim Auto-Probing.

Diese Reservierung ist beispielsweise für den IO-Bereich der NE2000-Ethernet-Karten sinnvoll, die nach einem unkontrollierten Zugriff durch einen wilden Gerätetreiber den Rechner blockieren können. Wenn Sie einen 32 Byte großen Adressbereich von hexadezimal `0x280` bis `0x29f` für die Netzwerkkarte (hier mit *IRQ* 10) reservieren wollen, lautet das

Kommandozeilenargument `reserve=0x280,32 ether=10,0x280,0,0,eth0`.

`mem=`**Bytes**

gibt die Größe des vorhandenen Arbeitsspeichers (RAM) an. Dieses Argument ist notwendig, wenn mehr als 64 Megabyte eingebaut sind. Beachten Sie, daß manchmal der oberste Speicherbereich als ``Shadow" vom BIOS belegt wird und deshalb für Linux nicht zur Verfügung steht!



Die Angabe der Speichergröße kann in Bytes, Kilobytes und Megabytes erfolgen. Die Einheiten müssen gegebenenfalls durch den nachgestellten Buchstaben k bzw. M gekennzeichnet werden.

Die folgenden Kommandozeilenargumente schalten bestimmte Kernelfunktionen an bzw. ab:

**debug**  
schaltet die Ausgaberoutine für Kernelmeldungen in den Debug-Modus.

**no387**  
schaltet den mathematischen Koprozessor ab. Damit der Kernel trotzdem startet, müssen die Routinen der Mathe-Emulation in den Kernel eingebunden sein.

**no-hlt**  
Mit dieser Option wird der Kernel veranlaßt, den Prozessorbefehl "hlt" zum Anhalten der CPU nicht auszuführen. Dieser Befehl führt bei einigen fehlerhaften Prozessoren dazu, daß das Betriebssystem "einfriert".

## RAM-Disk

**ramdisk\_size=Größe**

Durch dieses Kernelargument wird die Größe der Ramdisks festgelegt, die der Kernel anlegt, wenn der Gerätetreiber für dieses Gerät vorhanden ist. Der Kernel erzeugt die notwendigen Datenstrukturen um 16 Ramdisks der angegebenen Größe zu verwalten, der Arbeitsspeicher wird erst bei der tatsächlichen Belegung aus dem Buffer-Cache herausgenommen. Das obsoletere Schlüsselwort **ramdisk** wird bei Kernelversion 2.0 noch als Alias für **ramdisk\_size** akzeptiert, von der Verwendung wird aber abgeraten.

**load\_ramdisk={0,1}**

Mit diesem Argument wird bestimmt, ob der Kernel versucht eine RAM-Disk von Diskette zu laden (Wahrheitswert 1) oder nicht (Wahrheitswert 0).

**prompt\_ramdisk={0,1}**

Wenn dieses Kernelargument den Wahrheitswert 1 hat, wird der Bootvorgang zum Laden der RAM-Disk unterbrochen und der Operator erhält die Möglichkeit, die Diskette zu wechseln, von der die RAM-Disk gelesen wird.

**ramdisk\_start=BlockNr**

Mit dem Argument *BlockNr* kann dem Kernel mitgeteilt werden, in welchem Diskettenblock das Image der RAM-Disk anfängt.

## Festplatten und IDE-Controller

**xd=Typ,IRQ,IO-Port,DMA**

konfiguriert den Gerätetreiber für den 8-Bit-XT-Festplattencontroller.

**hd=Cyl,Head,Sect**

konfiguriert den Festplattentreiber für die normale (AT-Bus-) Festplatte /dev/hda mit den angegebenen Parametern.

**hdx=Cyl,Head,Sect**

konfiguriert den (E)IDE Treiber für den Betrieb von (E)IDE-Festplatten und ATAPI CD-ROM-Laufwerken wie z.B. NEC, Sony, Mitsumi und Versa. Der Parameter "hdx" kann für



jede Festplatte und jedes CD-ROM-Laufwerk einzeln angegeben werden. Das erste Gerät wird dabei mit `hda`, das zweite mit `hdb` usw. bezeichnet.

Anstelle des Tripels Zylinder,Kopf,Sektoren kann auch eines der Schlüsselwörter `none`, `noprobe`, `nowerr`, `cdrom`, `autotune` oder `noautotune` angegeben werden. Mit `none` und `noprobe` können einzelne Laufwerke vom Autoprobing ausgeschlossen werden. Mit `cdrom` wird ein ATAPI-CD-Laufwerk angezeigt. Durch die Schlüsselwörter `autotune` und `noautotune` kann der Treiber dazu veranlaßt werden, zusätzliche oder auch weniger Anstrengungen zur Performanceoptimierung zu unternehmen.

`iden=Base,Control,IRQ`

übergibt dem Treiber für den IDE-Controller die Hardwareparameter. Außerdem kann anstelle der Parameter eines der Schlüsselwörter `noprobe`, `autotune`, `noautotune`, `serialize`, `qđ6580`, `ht6560b`, `cmd640_vlb`, `dtc2278`, `umc8672`, `ali14xx` oder `dc4030` angegeben werden. Die drei ersten Schlüsselwörter haben die gleiche Bedeutung, wie bei den Treibern für die einzelnen IDE-Festplatten. Bei Systemen mit zwei IDE-Controllern wird der Kernel durch das Schlüsselwort `serialize` veranlaßt, die beiden Controller nicht gleichzeitig zu benutzen. Alle weiteren Schlüsselwörter zeigen dem Treiber einen bestimmten Chipsatz an, dessen spezielle Fehler (Features?) dadurch abgefangen werden können.

## SCSI-Kontroller

Einige SCSI-Hostadapter können durch die Angabe von IO-Port und IRQ auf der Kernelkommandozeile bei der Initialisierung unterstützt werden. Die Form des Kommandozeilenarguments ist `adapter=IO-Port,IRQ`. Als Adapter kommen `st0x` (Seagate ST0x), `tmx8xx` (TMC-885), `t128` (Trantor T128), `ncr53c400` (NCR53c400) und `dtc` (DTC 3180) in Frage.

Andere SCSI-Controller können mit weiteren Kommandozeilenargumenten konfiguriert werden:

`ncr5380=IO-Port,IRQ,DMA`

konfiguriert den generischen SCSI-Treiber für Hostadapter mit Chips der NCR 5380 Familie.

`aha152x=IO-Port,IRQ,SCSI-ID, Reconnect,Parity,Synchron, Delay`

konfiguriert den Gerätetreiber für den Adaptec 1520/1522 SCSI Controller. Diese Chips werden auch auf vielen Soundkarten mit SCSI-Unterstützung und auf anderen Billigcontrollern verwendet. Die Features *Reconnect* und *Parity* und *Synchron* können durch die Werte 0 oder 1 an- oder abgeschaltet werden, *Delay* hat die Vorgabe 100.

`aha1542=IO-Port[,BusOn, Busoff,[,DMA-Speed]]`

konfiguriert den Gerätetreiber für den Adaptec 1542 B/C(F) SCSI Hostadapter. *BusOn* ist die Zeit, die der Adapter den Bus für den Datentransfer belegt (2 bis 15 Mikrosekunden, Vorgabe 11), *BusOff* ist die Zeit, die der Adapter den Bus wieder freigibt (1 bis 64 Mikrosekunden, Vorgabe 4) und *DMA-Speed* ist die Transfargeschwindigkeit (5,6,7,8 oder 10 MB/s, Vorgabe 5).

`aic7xxx=extended,no_reset`

konfiguriert den Treiber für die Adaptec AIC-7xxx-basierten SCSI-Controller der Typen AHA-274x (EISA), AHA-284x (VLB), AHA-29xx (PCI), AHA-394x (PCI Twin) und AHA-398x (PCI RAID). Der AHA-2920 basiert nicht auf einem AIC-7xxx-Chip, er wird vom Future Domain Treiber (TMC-16x0) unterstützt.

Die Option `extended` schaltet die "Übersetzung" der Geometriedaten von Platten mit mehr als 1024 Zylindern ein. Mit `no_reset` wird das einige Sekunden dauernde Zurücksetzen des

SCSI-Bus beim Booten unterdrückt.

#### **in2000=Optionen**

konfiguriert den Always IN2000 SCSI-Controller. Der Treiber läßt sich über eine durch Komma getrennte Liste von Schlüsselwörtern oder Variablen einstellen:

##### **ioport:IO-Port**

setzt die Hardwareadresse des Controllers.

##### **noreset**

unterdrückt den Reset beim Booten.

##### **nosync:Maske**

Die Bits 1 bis 7 der Maske unterdrücken den synchronen Datentransfer zum Gerät mit der entsprechenden ID. Synchroner Transfer ist per Default ganz abgeschaltet.

##### **period:Nanosekunden**

ändert die minimale Transferdauer auf einen Wert zwischen 250 und 1000 ns.  
Voreinstellung ist 500ns.

##### **disconnect:Zahl**

mit 0 werden Disconnects niemals erlaubt, mit 2 immer.

##### **debug:Maske**

Wenn der Treiber mit DEBUG übersetzt wurde, legt die Maske fest, welche Informationen ausgegeben werden.

##### **proc:Maske**

Die Maske bestimmt, wie das Interface zum Proc-Dateisystem arbeitet.

#### **BusLogic=Optionen**

konfiguriert die meisten Controller der Familie der Buslogic SCSI-Kontroller. Die Liste der unterstützten Controller ist lang und die Optionen für die Kommandozeile entsprechend vielseitig. Als erstes Argument können Sie den IO-Port als Hexadezimalzahl angeben. Alle weiteren Optionen lesen Sie am besten in den Sourcen nach, eine Beschreibung würde zwei Seiten dieses Buches füllen.

#### **wd7000=IRQ,DMA**

konfiguriert den Treiber für den Western Digital WD-7000 Hostadapter mit den angegebenen Werten für Interrupt und DMA-Kanal.

#### **AM53C974=Host-ID,Target-ID,Rate, Offset**

konfiguriert den Treiber für AM53/79C974 SCSI-Controller. Durch die Kommandozeilenparameter wird der Datentransfer zwischen dem Host (angegeben durch seine SCSI-ID) und dem Gerät (Target-ID) eingestellt. Die Transferrate darf zwischen 3 und 10 liegen, der Sync-Offset zwischen 0 und 15.

#### **fdomain=IO-Port,IRQ,Adapter-ID**

konfiguriert den Treiber für den Future Domain Controller TMC-16x0 und Adaptec AHA-2920.

#### **pas16=IO-Port,IRQ**

konfiguriert den Pro Audio Spektrum 16 SCSI-Kontroller. Gibt man für den Interrupt den Wert 255 an, so wird der Kontroller ohne Interrupt betrieben. Es ist nicht möglich, den SCSI-Kontroller über den selben Interrupt wie die Soundkarte zu betreiben.

#### **advansys=IO-Port**

setzt die Hardwareadresse der SCSI-Hostadapter von AdvanSys.

**ppa=IO-Port,High,Low,NI**

konfiguriert den SCSI-Druckerport-Treiber für das Iomega ZIP-Drive. *High* ist die Verzögerung beim Datentransfer (Vorgabe 1us), *Low* ist die Verzögerung bei anderen Operationen (Vorgabe 6us). Mit Wert 1 für *NI* kann der Treiber in den 4-Bit-Modus geschaltet werden.

Bitte beachten Sie, daß der Parallelport, an dem das ZIP-Drive angeschlossen wird, nicht gleichzeitig als Druckerport konfiguriert werden darf. Um den Druckertreiber auszuschalten, können Sie zusätzlich folgenden Bootparameter angeben: `lp=0`

**max\_scsi\_luns=N**

gibt dem SCSI-Kontroller an, wieviele SCSI-Geräte maximal am SCSI-Bus angesprochen werden können. N muß zwischen eins und acht liegen.

## SCSI-Bandlaufwerke

**st=Puffergröße[,Schwelle[,MaxPuffer]]**

initialisiert den Treiber für SCSI-Bandlaufwerke. Die jedem Bandlaufwerk zugeteilte *Puffergröße* und der Schwellwert beim asynchronen Schreiben werden in Kilobyte angegeben. Wenn mehr als zwei Bandlaufwerke an einem Rechner betrieben werden sollen, kann *MaxPuffer* entsprechend hochgesetzt werden.

## CD-ROM-Laufwerke

Bei allen Treibern für eigenständige CD-ROM-Laufwerke kann mindestens der IO-Port durch ein Kommandozeilenargument auf dem Bootprompt eingestellt werden, bei weiteren Laufwerken ist zusätzlich noch der IRQ einstellbar. Die Form des Arguments `laufwerk=IO-Port[,IRQ]`. Treiber der einfachsten Form (nur IO-Port) sind: Sanyo (`sjcd`), Optics Storage (`optcd`) und GoldStar (`gscd`). Die zweite Form (mit IRQ) verstehen: Mitsumi (neu) (`mcdx`), Phillips CM206 (`cm206`) und Sony CDU535 (`sonycd535`).

Die anderen CD-Treiber verstehen zusätzliche Bootparameter:

**mcd=IO-Port,IRQ,Wait**

konfiguriert den alten Treiber für Mitsumi CD-ROM-Laufwerke. Mit dem Argument *Wait* kann der Timeout verlängert werden.

**cdu31a=IO-Port,IRQ[,PAS]**

konfiguriert Sony CDU 31/33A. Der Treiber führt kein Autoprobing durch, muß also immer durch einen Bootparameter konfiguriert werden. Die Angabe ``PAS" ist notwendig, wenn das CD-ROM über die Pro Audio Spektrum-Karte angeschlossen ist. Wird kein Interrupt für das Laufwerk verwendet, so muß der Wert 0 für IRQ angegeben werden.

**sbpcd=IO-Port,{SoundBlaster/LaserMate/SPEA}**

konfiguriert den Treiber für den Soundblaster Pro Multi-CD-Kontroller. Dieser Treiber sucht die ihm bekannten Laufwerke auf mehreren Ports, was sich anhand der Bootmeldungen mitverfolgen läßt.

Wenn Sie kein CD-Laufwerk dieses Typs eingebaut haben, können Sie die sehr zeitaufwendigen

Initialisierungsversuche des Treibers umgehen, indem Sie als Bootargument `sbpcd=off` angeben.

`aztcd=IO-Port,0x79`

konfiguriert den Aztech-CD-ROM-Treiber. Dieser erkennt Aztech, Orchid und Wearnes Laufwerke. Die Grundeinstellung ist `0x320`.

`isp16=IO-Port,IRQ,DMA,Typ`

konfiguriert den Treiber für OPTi 82C928/82C929 CD-Controller auf ISP16, MAD16 oder Mozart Soundkarten. Als *Typ* kommen folgende Schlüsselwörter in Frage: `noisp16`, Sanyo, Sony, Panasonic oder Mitsumi.

## Diskettenlaufwerke

Mit dem Schlüsselwort `floppy` kann dem Treiber für das Floppy-Laufwerk die Art und Anzahl der vorhandenen Diskettenkontroller bzw. Laufwerke mitgeteilt werden. Will man mehrere Optionen an den Diskettentreiber übergeben, so muß dieser Bootparameter mehrfach angegeben werden. Mögliche Optionen sind `thinkpad` für Besitzer eines IBM Thinkpads, `two_fdc`, wenn man zwei Diskettenkontroller verwenden will, und `all_drives`, wenn man an einem Diskettenkontroller mehr als die zwei per Default vorgesehenen Laufwerke betreiben möchte. Die Option `no_unexpected_interrupts` hilft bei manchen Laptops, die harmlosen, aber nervtötenden Meldungen `fd0: unexpected interrupt` abzustellen. Es gibt eine Reihe von weiteren Optionen für den Diskettentreiber, die bei Bedarf der Datei `drivers/block/README.fd` in den Kernelsourcen entnommen werden können.

## Verschiedenes

`lp=IO-Port[,IRQ,[IO-Port,IRQ...]]`

dient zur Konfiguration der parallelen Schnittstelle als Druckerport. Wenn dieses Argument auf dem Bootprompt auftaucht, findet kein Autoprobing für weitere Druckerports statt. Wenn ein Parallelport nicht für den Drucker sondern zum Beispiel für PLIP genutzt werden soll, kann durch Angabe von `lp=0` die Initialisierung des Druckertreibers verhindert werden.

`sound=Zahl[,Zahl...]`

konfiguriert den oder die Soundkartentreiber. Die *Zahl* wird in der Form `0xTaaaId` angegeben, wobei `T` für den Typ der Karte steht (1=FM Synth, 2=Soundblaster, 3=ProAudioSpektrum16, 4=Gravis Ultrasound 5=MPU-401, 6=Soundblaster 16, 7=Soundblaster 16 Midi). `aaa` steht für die hexadezimale IO-Adresse. `I` für den Interrupt (hexadezimal!) und `d` für den DMA-Kanal. Man kann dem Treiber auch mehrere Angaben gleichzeitig machen. Diese müssen dann durch ein Komma getrennt werden. Der Bootparameter könnte also folgendermaßen aussehen: `sound=222071,0x138800`.

## Busmäuse

Den Treibern für die Logitech-Busmaus und die Microsoft-Busmaus kann mit den Schlüsselwörtern `bmouse` (Logitech) und `msmouse` (Microsoft) jeweils der passende Hardware-Interrupt mitgeteilt werden.

# Netzwerkkarten

In der Form *ether=irq,IO-Port,Mem-Start, Mem-End,Device* wird der Treiber für die Ethernet-Karte mit den angegebenen Parametern initialisiert. Die Werte für den Beginn und das Ende des Shared-Memory-Bereiches sind eigentlich für Karten vorgesehen, die mit diesem Speicher arbeiten. Einige Netzwerkkartentreiber werten diese Parameter aus, obwohl sie kein Shared-Memory verwenden. Der Treiber für die AMD 79C960 Ethernetcontroller (Lance) benutzt den Wert von Mem-Start zur Festlegung des DMA-Ports, bei 3Com 3c501 und 3c507 wird durch die niedrigsten 4 Bit der Debug-Level des Treibers eingestellt. Der Treiber für den ATP-Adapter benutzt die unteren Bits von Mem-End zur Einstellung des Debug-Levels.

Die Bezeichnung für das Device wird im Normalfall `eth0` sein. Da der Kernel nach erfolgreicher Initialisierung einer Netzwerkkarte die Suche nach weiteren Karten aufgibt, kann nur durch explizite Beschreibung auf dem Bootprompt eine weitere Netzwerkkarte aktiviert werden.

Außer den hier aufgeführten Argumenten können noch weitere Optionen (Schalter) angegeben werden, die der Kernel an das *init*-Programm weiterreicht. Beispielsweise verstehen alle *init* Programme das Argument *single* als Befehl, das System im Einbenutzermodus zu starten.

Darüberhinaus können Gleichungen zur Definition von Umgebungsvariablen angegeben werden, die automatisch in der Prozeßumgebung aller laufenden Programme auftauchen.

## Die Vorgänge bei der Kernelinitialisierung

Wenn der Bootloader die Kerneldatei in den Arbeitsspeicher geladen hat, wird sie zunächst dekomprimiert und dann an die richtige Stelle (virtueller Adreßraum ab 3 GB) verschoben. Danach beginnt die zweite Phase der Systeminitialisierung, in der das eigentliche Betriebssystem startet und die unmittelbaren Hardwarekomponenten in Besitz nimmt.

Die Möglichkeiten für die Systemverwalterin, in dieser Phase in das Geschehen einzugreifen, sind minimal. Trotzdem werden die einzelnen Schritte hier einmal vorgestellt, damit Sie bei eventuell auftretenden Problemen den Fehler etwas besser eingrenzen können.

Zuerst werden einige interne Funktionen und Tabellen des Kernels initialisiert. Dazu gehören die Tabelle zur Speicherseitenverwaltung, die Belegung der Fehlerinterrupts und der IRQ, die Einrichtung der Prozeßtabelle und der Start der Schedulers. In dieser Phase wird auch die CMOS-Uhr ausgelesen und die Systemzeit entsprechend gesetzt. Wenn der Arbeitsspeicher Ihres Rechners ohne Fehler ist, sollte es hierbei keine Probleme geben. Es findet noch keine Bildschirmausgabe statt.

Dann wird die Kommandozeile des Kernels ausgewertet und die für den Kernel bestimmten Kommandozeilenargumente an die entsprechenden Setup-Funktionen übergeben. Später werden die in Kernel-Variablen gespeicherten Argumente bei der Initialisierung der Gerätetreiber verwendet. Die von der aktuellen Linux-Version unterstützten Kommandozeilenargumente sind im Abschnitt über den [Bootprompt](#) erklärt.

Damit alle weiteren Schritte auf dem Bildschirm verfolgt werden können, wird als nächstes die Systemconsole eingerichtet. Wenn für den Textbildschirm kein fester Video-Modus eingestellt wurde, wird an dieser Stelle eine Liste aller von der Grafikkarte unterstützten Modi ausgegeben und auf eine Auswahl gewartet.

Bei Geräten mit PCI-Bus wird als nächstes das PCI-Subsystem initialisiert. Dabei wird das PCI-BIOS aktiviert und der Bus nach Geräten abgesucht.

Danach wird die Geschwindigkeit des Rechners ermittelt und der Timer kalibriert. Die angezeigten *BogoMIPS* sind kein Benchmark und zum Vergleich der Rechnerleistung wenig geeignet.

Die Initialisierung des virtuellen Filesystems im Kernel findet normalerweise ``im Stillen" statt, so daß die nächste Bildschirmausgabe das Resultat der Speicherinitialisierung ist. Der Kernel zeigt hier normalerweise an, wieviel Arbeitsspeicher installiert ist und wieviel davon bereits vom Kernel belegt wurde.

Nach dem Speicher werden die geräteunabhängigen Netzwerkschichten initialisiert. Unter anderem werden auf dem Bildschirm die unterstützten IP-Protokolle angezeigt.


Im nächsten Schritt wird der Prozessor auf bekannte Fehler geprüft. Je nach Rechnerarchitektur und Prozessortyp können hier die Ergebnisse verschiedener Tests auf dem Bildschirm erscheinen.

### *Architektur & Test & Kommentar* Testroutinen für die verschiedenen Prozessoren

Intel x86	TLB	Wenn ein Kernel, der für Intel-Prozessoren 486 und höher übersetzt wurde, auf einem 386er Prozessor gestartet wird, kommt es zu einer Fehlermeldung und der Systemstart wird abgebrochen. Das Problem wird beseitigt, indem ein für den 386er Prozessor geeigneter Kernel geladen wird.
	HLT	Bei fehlerhaften Prozessoren kann es durch einen ``Halt"-Befehl zum Einfrieren des Systems kommen. Wenn es bei diesem Test zu Systemstillstand kommt, hat Ihr Prozessor diesen Fehler. Das Problem wird umgangen, indem der Kernel mit der Kommandozeilenoption <code>no-hlt</code> neu gestartet wird.
	FPU	Wenn der Rechner keinen mathematischen Koprozessor hat und der Kernel ohne FPU-Emulation übersetzt wurde, kommt es zu einer Fehlermeldung und der Systemstart wird abgebrochen. Das Problem wird beseitigt, indem ein anderer Kernel mit FPU-Emulation geladen wird. Bei älteren 386/387 Kombinationen teilt die FPU der CPU einen Fehler über den IRQ13 mit. Linux kann diesen Spezialfall erkennen und sich darauf einstellen. Im Normalfall löst ein Fehler die ``exception 16" aus. Bei Pentium-Prozessoren hat die Entdeckung eines Fehlers in der FPU für einiges Aufsehen gesorgt. Linux kann diesen Fehler (FDIV Bug) erkennen und umgehen.
MIPS	WAIT	Bei diesem Test wird festgestellt, ob der Prozessor die Instruktion ``wait" versteht.
ALPHA	--	Zur Zeit sind keine Bugs des ALPHA-Prozessors bekannt.
68k	--	Es gibt keine Fehlerbehandlung für Prozessoren dieses Typs.
SPARC	--	Es gibt keine Fehlerbehandlung für Prozessoren dieses Typs.
PowerPC	--	Es gibt keine Fehlerbehandlung für Prozessoren dieses Typs.

Nach dem Prozessortest wird als letzter Schritt in der zweiten Phase der Systeminitialisierung der ``Linux-Banner" ausgegeben, in dem die Versionsnummer und das Übersetzungsdatum des Kernels angezeigt werden.

Die dritte Phase der Initialisierung beginnt, indem der Kernel einen ``*Kernel-Thread*" mit dem Namen `init` erzeugt. Ein Kernel-Thread ist eine Kernelfunktion, die bereits vom Scheduler verwaltet wird wie ein Prozeß, sich aber den Speicherbereich mit den anderen Kernelfunktionen teilt.

init erzeugt zunächst zwei weitere Kernel-Threads, bdf flush  und kswapd, die den Kernel bei der Verwaltung des virtuellen Speichers und des Buffer-Cache unterstützen. Danach wird der Systemcall setup ausgeführt, der für die Initialisierung der Gerätetreiber, des Programmladers und der Dateisysteme verantwortlich ist.

Die Kernelfunktion setup ruft einen Gerätetreiber nach dem anderen auf und veranlaßt ihn, nach "seinem" Gerät zu suchen, es zu initialisieren und sich nach einer erfolgreichen Initialisierung im Kernel zu registrieren. Manche Gerätetreiber geben genaue Statusinformationen und schreiben eine Erfolgsmeldung auf den Bildschirm. In der Tabelle 5.2 sind diese Gerätetreiber mit dem Buchstaben **E** wie "Erfolg" gekennzeichnet. Die meisten Gerätetreiber geben wenigstens eine Fehlermeldung aus, wenn etwas bei der Initialisierung schiefgegangen ist. Diese Treiber sind mit einem **F** wie "Fehler" gekennzeichnet. Treiber, die weder mit E noch mit F gekennzeichnet sind, arbeiten völlig stumm.

Es sind nicht in jedem Kernel alle möglichen Gerätetreiber enthalten. Viele Treiber werden nur benötigt, wenn die entsprechende Hardware installiert ist. Solche Treiber sind "optional", sie werden bei der Konfigurierung des Kernels vor dem Übersetzen der Sourcen angeboten und können ausgewählt oder auch weggelassen werden. Die meisten dieser optionalen Treiber können auch zu einem Modul gemacht werden, das nicht dauerhafter Bestandteil des Kernels ist, sondern bei Bedarf zum laufenden Betriebssystem hinzugeladen werden kann. Optionale Treiber, die zu einem Kernelmodul gemacht werden können, sind in der Tabelle mit einem **M** gekennzeichnet, die anderen mit einem **O** wie "Option".

Die Tabelle listet alle in den Sourcen enthaltenen Treiber in der Reihenfolge ihrer Initialisierung auf. Wenn es beim Starten des Systems zu Fehlern kommt oder wenn Ihre Hardware nicht korrekt erkannt wird, kann Ihnen diese Information bei der Eingrenzung des Problems helfen.

Um das Problem zu beheben, können Sie häufig durch Kommandozeilenoptionen, die dem Kernel vor dem Laden mit dem Bootprompt übergeben werden, Einfluß auf den oder die problematischen Treiber nehmen. In der dritten Spalte der Tabelle ist zu allen Gerätetreibern, die mit einem Bootparameter von der Kernelkommandozeile beeinflußt werden können, das Kommandozeilenargument in Kurzform dargestellt. Sie finden eine ausführliche Beschreibung im Abschnitt über die [Bootparameter](#).

Gerätetreiber&Info&Setup-Parameter  
Initialisierung der Gerätetreiber

3[c Zeichenorientierte Geräte		
Zufallszahlengenerator	--	
Tastatur	F	
Serielle Schnittstellen	EFM	
Z8530 basierte HDLC Karte (SCC)	EFM	
Cyclades Multiport	EFM	
Stallion Multiport	EFM	
Stallion "intelligenter" Multiport	EFM	
DigiBoard PC/Xe Multiport	EFO	digi=

RISCom/8 Multiport	EFM	<i>riscom8=IO-Port</i>
Baycom Funkmodems	EFM	<i>baycom=Modem,IO-Port,IRQ,DCD</i>
Pseudoterminals	F	
Virtuelle Console Screen	F	
Drucker	EFM	<i>lp=IO-Port,IRQ</i>
Logitech Busmaus	EM	<i>bmouse=IRQ</i>
PS/2 Busmaus	EFM	
Microsoft BusMouse	EM	<i>msmouse=IRQ</i>
ATI Busmaus	EM	
Watchdog Timer	EFM	
Advanced Power Management	EFO	
Echtzeituhr	EFO	
Soundkarte	EFM	<i>sound=0xTaaaId</i>
QIC-02 Tape	EFO	
ISDN Subsystem	EFM	
ICN ISDN Karte	EFM	<i>icn=IO-Port,SharedMem</i>
Teles ISDN Karte	EFM	<i>teles=IO-Port,IRQ,SharedMem,Prot</i>
PCBIT ISDN Karte	EFM	<i>pcbit=IO-Port,IRQ</i>
Ftape	EFM	
3[c]Blockorientierte Geräte		
Ram-Disk	EFM	<i>ramdisk_size=Größe</i>
Loop Filesystem Driver	EFM	
ISP-16 CD	EFM	<i>isp16=IO-Port,IRQ,DMA,Typ</i>
IDE-Interface	EFO	<i>idex=Optionen</i>
IDE-Geräte	EFO	<i>hdx=Optionen</i>
IDE/MFM/RLL Festplatten	FO	<i>hd=Cyl,Head,Sec</i>
XT-Festplatten	FM	<i>xd=Typ,IRQ,IO-Port,DMA</i>
Diskettenlaufwerke	EFM	<i>floppy=Optionen</i>
Sony CDU-31A	EFM	<i>cdu31a=IO-Port,IRQ[,Typ]</i>
Mitsumi CD	EFM	<i>mcd=IO-Port,IRQ,Wait</i>
Mitsumi Multisession	EFM	<i>mcdx=IO-Port,IRQ</i>



SBP CD	EFM	<i>sbpcd=IO-Port,Typ</i>
Aztech CD	EFM	<i>aztcd=IO-Port,Maggi</i>
Sony CDU-535	EFM	<i>sonycd535=IO-Port,IRQ</i>
GoldStar	EFM	<i>gscd=IO-Port</i>
Phillips CM206	EFM	<i>cm206=IO-Port,IRQ</i>
Optics Storage	EFM	<i>optcd=IO-Port</i>
Sanyo CD	EFM	<i>sjcd=IO-Port</i>
Multiple Device	EFM	
3 c SCSI Komponenten		
Amiga 3000	EFM	
Commodore A2091	EFM	
GVP Serie II	EFM	
Atari Native	EFO	
AdvanSys	EFM	<i>advansys=IO-Port</i>
BusLogic	EFM	<i>BusLogic=Optionen</i>
UltraStor 14F/34F	EFM	
UltraStor 14F	EFM	
Adaptec AHA152x	EFM	<i>aha152x=Port,IRQ,ID,Recon,Par</i>
Adaptec AHA1542	EFM	<i>aha1542=Port,BusOn,BusOff,Rate</i>
Adaptec AHA1740	EFM	
Adaptec AIC7xxx	EFM	<i>aic7xxx=extended,no_reset</i>
Future Domain 16x0		<i>fdomain=IO-Port,IRQ,Adapter-ID</i>
Always IN2000	EFM	<i>in2000=Optionen</i>
NCR5380 Generic	EFM	<i>ncr5380=IO-Port,IRQ,DMA</i>
NCR53c400	EFM	<i>ncr53c400=IO-Port,IRQ</i>
NCR53c406a	EFM	<i>ncr53c406a=IO-Port,IRQ,Fast-PIO</i>
Qlogic FAS408	EFM	
Pro Audio Spectrum	EFM	<i>pas16=IO-Port,IRQ</i>
Seagate ST0x	EFM	<i>st0x=IO-Port,IRQ</i>
Future Domain TMC-885, TMC-950	EFM	<i>tmc8xx=IO-Port,IRQ</i>
Trantor T128	EFM	<i>t128=Memory,IRQ</i>

DTC 3180/3280	EFM	(d <sub>tc</sub> = <i>Memory</i> , <i>IRQ</i> )
NCR 53C700/53C700-66	EFM	
NCR 53C810	EFM	
EATA HBA	EFM	
EATA PIO	EFM	
Western Digital WD-7000	EFM	wd7000= <i>IRQ</i> , <i>DMA</i>
EATA/DMA 2.0x	EFM	
AM53C974	EFM	AM53C974= <i>Host</i> , <i>Targ</i> , <i>Rate</i> , <i>Offset</i>
QLogic ISP1020	EFM	
Iomega ZIP (Parallel Port)	EFM	ppa= <i>IO-Port</i> , <i>High</i> , <i>Low</i> , <i>N1</i> , <i>N2</i>
Gerätescan		max_scsi_luns= <i>Anzahl</i>
SCSI Tape	EFM	st= <i>Buffer</i> , <i>Schwelle</i> , <i>MaxBuf</i>
3c Netzwerk		
Ethernetkarten	EFM	ether= <i>IRQ</i> , <i>IO-Port</i> , <i>MemStart</i> , <i>MemEnd</i>

Nachdem alle erkannten Geräte beim Kernel angemeldet sind, werden die Partitionen und die Größe der Festplatten geprüft, eventuell vorgesehene RAM-Disks in den Speicher gelesen, die Kernelfunktionen zum Laden der verschiedenen Binärformate vorbereitet und schließlich die unterstützten Dateisysteme registriert und das Rootfilesystem gemountet.

An dieser Stelle ist die dritte Phase der Systeminitialisierung abgeschlossen. Das Betriebssystem hat die Kontrolle über die Komponenten und Geräte des Systems und ist bereit, Arbeitsprozesse in einem jeweils eigenen ``Userspace" zu starten.

## RAM-Disk und Initrd

Sie sind für die Installation oder Reparatur eines festplattenbasierten Linux-Systems wichtig, tauchen aber beim alltäglichen Betrieb von Linux selten wieder auf: die RAM-Disks.

Wenn der Kernel den Treiber für die RAM-Disks enthält, werden bei der Initialisierung der Blockgeräte die Verwaltungsstrukturen für 16 dieser ``Geräte" angelegt. Wenn mit dem Bootprompt keine andere Größe eingestellt wurde, kann jede dieser RAM-Disks bis zu 4MB groß werden. Der Arbeitsspeicher wird nicht bei der Initialisierung, sondern erst beim Beschreiben der RAM-Disk belegt.

Wenn eine RAM-Disk nach Abschluß der dritten Phase des Bootvorgangs von einer Floppy in den Speicher geladen werden soll, müssen dem Kernel die Details der RAM-Disk über Kommandozeilenargumente auf dem Bootprompt mitgeteilt werden.

1.

Durch den Bootparameter `ramdisk_size=NNNN` wird die maximale Größe *aller* RAM-Disks

verändert. Die Angabe erfolgt in Kilobytes. Dieser Wert muß nicht mit der tatsächlich benutzten Größe übereinstimmen, darf jedoch nicht kleiner sein. Die Vorgabe von 4096kB (4MB) ist in der Regel ausreichend. Weil der Speicher nur bis zur tatsächlich benutzten Größe belegt wird, ist eine Verkleinerung nicht sinnvoll.

2.

Durch den Bootparameter `load_ramdisk=1` wird der Kernel überhaupt erst veranlaßt, auf der Floppy nach einer ladbaren RAM-Disk zu suchen, um sie gegebenenfalls in den Speicher zu lesen.

3.

Wenn der Bootparameter `prompt_ramdisk=1` vorhanden ist, ermöglicht der Kernel das Wechseln der Diskette, von der die RAM-Disk geladen wird.

4.

Mit dem Bootparameter `ramdisk_start=NNN` kann die Position des RAM-Disk-Image auf der Diskette angegeben werden. Diese Information ist notwendig, wenn nicht der gesamte Disketteninhalt (beginnend bei Block 000) als RAM-Disk geladen werden soll. Mit dem Argument `NNN` wird die Nummer des Diskettenblocks angegeben, aus dem der Anfang der RAM-Disk geladen werden soll.

Das Image der RAM-Disk kann auch komprimiert sein, der Kernel erkennt das automatisch und dekomprimiert es beim Lesen.

Die Werte für `load_ramdisk`, `prompt_ramdisk` und `ramdisk_start` können durch das Systemprogramm [rdev](#) auch fest in das Kernelimage eingetragen werden.

Ein kurzes Beispiel soll die Erzeugung einer RAM-Disk und einer Bootdiskette zum Laden derselben zeigen:

```
[01] # dd if=/dev/zero of=/dev/ram bs=1k count=2048
[02] # mke2fs -m0 /dev/ram 2048
[03] # mount /dev/ram /mnt/ramdisk
[04] # cp -a /tmp/myramdisk/* /mnt/ramdisk
[05] # umount /mnt/ramdisk
[06] # dd if=/dev/ram bs=1k count=2048 | gzip -v9 > /tmp/ram_image.gz
[07] # dd if=/vmlinuz of=/dev/fd0 bs=1k
[08] # dd if=/tmp/ram_image.gz of=/dev/fd0 bs=1k seek=500
[09] # rdev /dev/fd0 /dev/fd0
[10] # rdev -r /dev/fd0 16884
[11] # _
```

Durch das erste Kommando werden 2048 Nullbytes in die RAM-Disk geschrieben und dadurch 2MB aus dem Buffercache belegt. Dieser Speicherbereich gehört von nun an der RAM-Disk. Im zweiten Kommando wird in der neuen RAM-Disk ein Dateisystem angelegt. Die Kommandos 3-5 dienen dazu, die RAM-Disk mit sinnvollen Daten zu füllen. Im 6. Kommando wird das Image der RAM-Disk auf Festplatte geschrieben und gleichzeitig komprimiert. Dann wird der Kernel auf die Diskette kopiert und anschließend das komprimierte Image der RAM-Disk dahinter geschrieben. Die letzten beiden Kommandos veranlassen den Kernel, die RAM-Disk aus dem Block 500 zu laden und als Rootfilesystem zu mounten. Natürlich darf der Kernel nicht größer als 500kB und das Image der RAM-Disk maximal 940kB groß sein, damit dieses Beispiel funktioniert.

Speziell für die Konstrukteure der mehr oder weniger genialen Installationsprozeduren der

verschiedenen Linux-Distributionen gibt es die Möglichkeit, eine sogenannte Init-RAM-Disk (*initrd*) zu laden. Die wird gegebenenfalls nach erfolgreichem Setup noch vor dem eigentlichen Root-Filesystem gemountet und ein darauf befindliches Programm oder Shellsript `linuxrc` abgearbeitet. Dieses Programm läuft im Userspace und kann zum Beispiel dafür sorgen, daß die passenden Kernelmodule für bestimmte Komponenten wie CD-ROM oder SCSI-Host nachgeladen werden.

Wenn das Programm auf der Initrd beendet ist, wird das "richtige" Root-Filesystem gemountet und die Init-RAM-Disk auf das Verzeichnis `/initrd` verschoben.

Genauere Informationen zur Verwendung der Initrd finden Sie bei den Kernelsourcen in der Datei `./linux/Documentation/initrd.txt`.

## init

Wenn die Setup-Phase der Systeminitialisierung abgeschlossen ist und eventuell notwendige Treiber von der Initrd nachgeladen wurden, hat der Kernel schließlich eine feste Verbindung zu einem Dateisystem (Rootfilesystem) und kann von dort Programmdateien laden und ausführen.

Die vierte und letzte Phase der Systeminitialisierung beginnt, wenn der zunächst im Kernelspeicher gebliebene *init*-Thread die Programmdatei `/sbin/init` in den Arbeitsspeicher lädt und ausführt. Durch diesen Vorgang wird der erste Prozeß im Userspace erzeugt. Der Prozeß unterscheidet sich vom Thread dadurch, daß er einen eigenen virtuellen Adressraum benutzt, der einen vom Kernelspeicher verschiedenen Bereich des Arbeitsspeichers belegt. Der Prozeß *init* ist verantwortlich für die Entstehung aller weiteren Prozesse. Außerdem sorgt *init* für eine gewisse Minimalausstattung der Arbeitsumgebung.

Typischerweise führt *init* zu Beginn seiner Laufzeit ein oder mehrere Shellprogramme (Batchdateien) aus. Dadurch werden, ähnlich wie mit `autoexec.bat` bei DOS, bestimmte Einstellungen automatisch vorgenommen. Die Multitaskingfähigkeit von Linux ermöglicht auch den Start von parallel laufenden Serviceprogrammen, sogenannten Dämonen, die dem Systembenutzer jederzeit ihre Dienste anbieten.

Es ist charakteristisch für den Prozeß *init*, daß er nicht beendet wird. Auf diese Weise sorgt *init* dafür, daß bestimmte Prozeduren, vor allem das Login auf den virtuellen oder realen Terminals, in immer neuen Zyklen gestartet werden.

Für Linux sind zwei sehr verschiedene Versionen von *init* verbreitet:

Das einfache *init* (*simpleinit*) von Peter Orbaek mit Erweiterung von Werner Almesberger beschränkt sich auf die wesentlichen Aufgaben: es wird ein einziges Shellsript, die Datei `/etc/rc`, der Standardshell zur Interpretation übergeben, und es werden die in der Datei `/etc/inittab` bestimmten virtuellen Terminals und seriellen Schnittstellen mit einem *getty*-Prozeß belegt. Für spezielle Situationen kann das *simpleinit* im Einbenutzermodus (single user mode) gestartet werden. In diesem Modus wird das Initialisierungsscript nicht interpretiert, und es wird nur eine einzige Shell mit Root-Rechten gestartet.

Immer häufiger wird das vielseitigere System-V-kompatible *sysvinit* von Mike Jagdis und Miquel

van Smoorenburg installiert. Mit `sysvinit` können zu den beiden Systemzuständen ``*Singleuser*'' und ``*Multiuser*'' noch feinere Unterteilungen in sogenannte *Runlevel* vorgenommen werden. Wie beim `simpleinit` werden auch vom `sysvinit` die in der `inittab` bestimmten Login-Prozeduren gestartet und zusätzlich verschiedene Programme oder Shellscripts zur Initialisierung der *Runlevel* abgearbeitet.

Wenn Sie nicht sicher sind, welches `init` bei Ihrem Linux-System installiert ist, können Sie die Textdatei `/etc/inittab` mit den Beispielen aus dem nächsten Abschnitt vergleichen und damit Ihr `init` einer der beiden Versionen zuordnen.

## Die Konfigurationsdatei `/etc/inittab`


Bei beiden Versionen von `init` wird der genaue Ablauf der letzten Initialisierungsphase zunächst durch die Konfigurationsdatei `/etc/inittab` festgelegt. Die Form dieser Datei ist für die beiden Programmversionen verschieden. Wenn Sie die Konfigurationsdatei verändern, müssen Sie sorgfältig darauf achten, daß Sie die korrekte Syntax für die bei Ihnen installierte Version verwenden.

Ein grober Fehler in der `inittab` kann dazu führen, daß die Systeminitialisierung hängen bleibt und kein Login möglich ist. In solch einem Fall können Sie sich helfen, indem Sie beim LILO-Bootprompt die Kommandozeilenoption `init=/bin/sh` angeben. Auf diese Weise wird die Shell anstelle von `init` geladen und Sie haben die Möglichkeit, das System zu reparieren.

Beim `simpleinit` wird in der `inittab` nur festgelegt, auf welchen Terminals und mit welchen Parametern ein `getty` gestartet wird, um dort ein Login zu ermöglichen. Die Einträge in der `inittab` für `simpleinit` bestehen aus drei Feldern, die durch Doppelpunkte getrennt werden:

*Terminal:Termcapeintrag:Gettykommando*

Der erste Eintrag bezeichnet das Terminal (`tty1`, `tty2`, `ttyS1` ...), wie es in die `/var/run/utmp`

 Datenbank eingetragen und wie es vom `who` Kommando angezeigt wird. Der zweite Eintrag wird automatisch in die `TERM` Umgebungsvariable der auf dem entsprechenden Terminal gestarteten Shell geschrieben und sollte deshalb mit einem Eintrag in der `/etc/termcap` Datei übereinstimmen.


Der dritte Eintrag schließlich ist die Kommandozeile, mit der das `getty`-Kommando für das Terminal aufgerufen werden muß. Da es auch vom `getty`-Programm verschiedene Versionen gibt, sollte das entsprechende Format am besten der Beschreibung dieses `getty` entnommen werden.

```
# Beispiel einer inittab fuer simpleinit.
# Format: tty-Port:termcap-Eintrag:getty-Kommando
#
# Die ersten fuenf Zeilen sind fuer die virtuellen Terminals...
#
tty1:linux:/sbin/getty 9600 tty1
tty2:console:/sbin/getty 9600 tty2
tty3:console:/sbin/getty 9600 tty3
tty4:console:/sbin/getty 9600 tty4
#tty5:con100x40:/sbin/getty 9600 tty5
#
# ...die letzte Zeile ist fuer den Modem-Port.
#
ttyS1:vt102:/sbin/getty 9600 ttyS1
```

Das Verfahren, mit dem `simpleinit` die Kommandos aus der `inittab` bearbeitet, entspricht ungefähr der Methode `respawn` vom `sysvinit`.

Das System-V-kompatible `init` (**`sysvinit`**) hat auch vor allem die Aufgabe, die `gettys` für die Login-Prozeduren zu erzeugen und immer neu zu starten. Die Verwaltung zusätzlicher Runlevel erfordert zusätzliche Einträge mit einer anderen Syntax in der `inittab`. Jede Zeile, die nicht mit einem ``#'` (Kommentar) beginnt, ist ein Datensatz mit vier durch Doppelpunkt voneinander getrennten Einträgen:

*ID:Runlevel:Aktion:Prozeß*

Der erste Eintrag ist eine vierstellige  Zeichenkette als Bezeichner für die Zeile. Die ID wird bei Einträgen in Logfiles und bei Fehlermeldungen ausgegeben.

Für die Einträge, mit denen Login-Prozeduren auf den Terminals gestartet werden sollen, sollte die ID mit der charakteristischen Namensendung der entsprechenden Gerätedatei übereinstimmen, damit beim Accounting die Daten korrekt ausgewertet werden können. Beispielsweise soll die ID `S1` für ein `getty` auf `ttyS1` verwendet werden.

Im zweiten Eintrag werden die Runlevel festgelegt, in denen die entsprechende Zeile ausgewertet werden soll. Die Runlevel werden mit Ziffern von 0 bis 9 und mit den Buchstaben A, B, C, Q oder S (ohne Unterscheidung zwischen Groß und Kleinschreibung) bezeichnet. Einer Zeile können bis zu 11 Runlevel zugeordnet werden. Bei den Buchstabenkennungen für die Level steht S für den Einbenutzermodus, Q für Quit zum Neueinlesen der `inittab`. Die anderen Buchstaben werden für ondemand Aufrufe verwendet, bei denen ein Kommando beim Moduswechsel nicht mehr abgebrochen wird.

Wenn das Feld für den Runlevel leer ist, wird die Aktion bei jedem Moduswechsel ausgeführt.

Im dritten Eintrag wird bestimmt, welche Aktion von `init` ausgeführt wird:

`initdefault`

Das zweite Feld dieser Zeile bestimmt den Runlevel beim Systemstart. Dieser Eintrag kann durch einen entsprechenden Parameter auf der Kommandozeile (wie er z. B. von LILO übergeben wird) ersetzt werden.

`sysinit`

Das Kommando im vierten Feld dieser Zeile wird einmal unmittelbar nach dem Systemstart ausgeführt, noch bevor irgendein Anwender Zugang zum System erhält, also auch vor dem Einbenutzermodus.

`bootwait`

Das Kommando dieser Zeile wird einmal ausgeführt, wenn in einem Mehrbenutzermodus gestartet oder vom Einbenutzermodus dorthin gewechselt wird. `init` wartet mit der Bearbeitung weiterer Zeilen aus der `inittab`, bis das in dieser Zeile aufgerufenen Kommando beendet ist.

`boot`

Das Kommando im vierten Feld dieser Zeile wird wie bei `bootwait` ausgeführt, jedoch wird sofort mit der Bearbeitung der `inittab` fortgefahren, ohne auf die Beendigung des Kommandos zu warten.

`respawn`

Das Kommando im vierten Feld dieser Zeile wird beim Übergang in einen im zweiten Feld

aufgeführten Modus gestartet (wenn es nicht bereits läuft), und es wird jedesmal neu gestartet, wenn es beendet wurde. Das bedeutet, daß `init` jedes mit `respawn` gestartete Kommando dauernd überwacht. Wenn das Kommando normal beendet oder während der Laufzeit durch ein Signal terminiert wird, wird es sofort wieder neu gestartet. Wenn ein mit `respawn` gestartetes Kommando nicht stabil läuft und deshalb mehr als 10 mal in zwei Minuten neu gestartet werden muß, setzt `init` das Kommando für die Dauer von fünf Minuten aus.

#### ondemand

hat die gleiche Funktionalität wie `respawn` und wird benutzt, um im Zusammenhang mit den durch Buchstaben gekennzeichneten Leveln einzelne Kommandos „auf Anfrage“ mit `telinit` zu starten.

#### wait

Das Kommando in dieser Zeile wird beim Übergang in einen im zweiten Feld aufgeführten Modus ausgeführt, mit der Bearbeitung weiterer Zeilen der `inittab` wird gewartet, bis das Kommando beendet wurde.

#### once

Das Kommando im vierten Feld dieser Zeile wird beim Übergang in einen passenden Modus einmal aufgerufen. Auf die Beendigung wird nicht gewartet.

#### off

Wenn das Kommando im vierten Feld läuft, wird es angehalten, sonst wird der Eintrag ignoriert.

#### ctrlaltdel

Der Linux-Kernel kann aus der Tastenkombination CONTROL-ALT-DELETE ein Interrupt-Signal für `init` generieren und damit ein Reboot auslösen. Durch den mit `ctrlaltdel` gekennzeichneten Eintrag in der `inittab` wird das Kommando festgelegt, das zum Rebooten des Systems benutzt werden soll.

#### kbrequest

Diese Aktion wird ausgelöst, wenn `init` das Signal SIGWINCH von der Tastatur erhält.

#### powerwait

Das Kommando aus dem vierten Feld wird ausgeführt, wenn `init` sich in einem passenden Runlevel befindet und das Signal SIGPWR erhält oder über die Pipeline `/dev/initctrl` den entsprechenden Befehl erhält. `init` wartet auf die Beendigung des Kommandos. Das Signal oder der Befehl in der Pipeline kann mit Hilfe eines Dämons von einer USV (Unterbrechungsfreie Stromversorgung) erzeugt werden, wenn die Netzspannung abgefallen ist.

#### powerfail

Das Kommando dieser Zeile wird wie bei `powerwait` ausgeführt, `init` wartet jedoch nicht auf dessen Beendigung.

#### powerfailnow

Das Kommando wird unter den gleichen Bedingungen wie `powerwait` ausgeführt. Wenn es durch ein Signal ausgelöst wird, muß die Datei `/etc/powerstatus` zusätzlich das Wort `Low` enthalten. Auf diese Weise kann das Advanced Power Management eines Notebooks mit Hilfe eines Dämons das System automatisch herunterfahren, wenn die Batteriespannung ihren Minimalwert erreicht hat.

#### powerokwait

Das Kommando im vierten Feld wird unter den gleichen Bedingungen wie `powerwait` ausgeführt. Wenn es durch ein Signal ausgelöst wurde, muß die Datei `/etc/powerstatus`

zusätzlich das Wort OK enthalten. Auf diese Weise kann das System in den normalen Zustand gebracht werden, nachdem die Stromversorgung wieder hergestellt ist.

```
# Beispiel einer inittab fuer sysvinit
# Format: ID:Runlevel:Aktion:Kommando

# Einmalige Systeminitialisierung noch vor dem Einbenutzermodus:
si::sysinit:/etc/rc.d/rc.sysinit

# Der Eintrag fuer den Einbenutzermodus:
~~:S:wait:/sbin/sulogin

# Der Default-Runlevel:
id:2:initdefault:

# Einmalige Vorbereitung des Mehrbenutzermodus durch eine Batchdatei:
bo::bootwait:/etc/rc.d/rc.boot

# Bei jedem Wechsel in einen Runlevel 2 bis 5 wird das Shellscript
# /etc/rc.d/rc aufgerufen, um die weitere Arbeit zu erledigen:
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5

# Der "Affengriff" fuer den Warmstart:
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -r now

# Schließlich die Login-Prozeduren:
1:2345:respawn:/sbin/getty 9600 tty1
2:23:respawn:/sbin/getty 9600 tty2
3:23:respawn:/sbin/getty 9600 tty3
4:23:respawn:/sbin/getty 9600 tty4
```

Bei der Verarbeitung der inittab geht sysvinit folgendermaßen vor:

1.

Wenn `init` nicht das Argument `emergency` vom LILO Bootprompt erhalten hat, werden alle Kommandos mit der Aktion `sysinit` der Reihe nach abgearbeitet.

2.

Wenn `init` nicht über die Kommandozeile vom Bootprompt einen Runlevel als Argument erhalten hat, wird in der `inittab` ein Eintrag für `initdefault` gesucht. Wenn solch ein Eintrag nicht existiert, erfragt `sysvinit` den Runlevel von der Systemconsole. Falls `/dev/console` nicht geöffnet werden kann, startet `sysvinit` automatisch im Single-User-Modus.

3.

Falls als System im Einbenutzermodus gestartet wird, führt `sysvinit` der Reihe nach alle



Zeilen für den Runlevel `S` aus. Wenn in der `inittab` keine solche Zeile existiert, wird automatisch eine Shell mit der Aktion `wait` gestartet.

Wenn das letzte Kommando im Einbenutzermodus beendet ist, fährt `sysvinit` mit dem Booten eines Mehrbenutzermodus fort.

4.

Bevor der endgültige Runlevel erreicht wird, wechselt `sysvinit` zunächst in den Bootlevel und führt alle Zeilen mit den Aktionen `boot` und `bootwait` aus.

5.

Schließlich werden alle Zeilen aus der `inittab` der Reihe nach abgearbeitet, die für den als Ziel bestimmten Runlevel vorgesehen sind.

6.

Bei jedem weiteren Wechsel des Runlevel, wie er durch das Programm `telinit` erzielt werden kann, werden überflüssige noch laufende Kommandos beendet und eventuell notwendige neue Kommandos gestartet.

## Respawn mit Initscript

Wenn ein Shellsript mit dem Namen `/etc/initscript` existiert, werden die Kommandos der Aktionen `ondemand` und `respawn` durch dieses Script ausgeführt. Auf diese Weise können bestimmte Einstellungen an der Prozeßumgebung vorgenommen werden, die an alle abzweigenden Prozesse vererbt werden.

Das Script wird von `init` in der folgenden Form aufgerufen:

```
/etc/initscript ID Runlevel Aktion Kommando
```

Auf die Argumente kann in dem Shellsript über die Positionsparameter zugegriffen werden.

Ein kurzes Beispiel zeigt, wie das eigentliche Kommando aus dem `initscript` gestartet wird:

```
# Beispiel fuer /etc/initscript
# Zuerst werden die gewuenschten Einstellungen vorgenommen,...
umask 022
ulimit -S -m 2048 -c 4096

# ...danach wird das eigentliche Kommando ausgefuehrt.
eval exec "$4"
```

## Die Shellscrip te zur Systeminitialisierung

Wenn Sie Ihr Linux-System im Einbenutzermodus oder ganz ohne `init` mit einer einfachen Shell starten, finden Sie die Laufzeitumgebung vor, wie ein weißes Blatt Papier. Sie können sich Stück für Stück das gewünschte Bild schaffen, indem Sie die entsprechenden Elemente einzeln ``von Hand" hinzufügen.

Um sich die gleiche Arbeit beim nächsten Systemstart zu sparen, können Sie die Kommandos, die Sie zur Herstellung Ihrer Arbeitsumgebung eingeben müssen, in eine Datei schreiben. Indem Sie diese Datei zu Beginn jeder Session als Shellsript ausführen, erzeugen Sie nach jedem Systemstart das

gewohnte Anfangsbild.

`simpleinit` führt genau ein solches Shellsript automatisch beim Start in den Mehrbenutzermodus aus: die Datei `/etc/rc`. Der Name `rc` kommt von *run command* (*runcom*). Die Rolle dieser Datei ist vergleichbar mit der `AUTOEXEC.BAT` bei DOS.

```
# Beispiel einer /etc/rc fuer simpleinit

# Das Root-Filesystem ist Read-Only gemountet.
# Zuerst werden die Dateisysteme getestet...
/sbin/fsck -A -a
if [ $? -gt 1 ] ; then
    echo "Fehler im Dateisystem gefunden. Bitte neu booten!"
    /bin/sh
fi

# ...dann wird das Root-Filesystem mit Schreiberlaubnis remountet.
/sbin/mount -n -o rw,remount /

# Hier werden zur Sicherheit ein paar Dateien entfernt.
/bin/rm -f /etc/mtab* /var/lock/LCK* /var/run/*
/bin/cat /dev/null > /etc/utmp

# Hier wird der Rest des Dateisystems ohne das NFS gemountet
# und die Swappartition aktiviert.
/sbin/mount -avt nonfs
/sbin/swapon -a

# Einige Daemonen werden gestartet.
/usr/sbin/crond
/usr/sbin/lpd
# Der update Daemon muss unbedingt im rc-Script gestartet werden.
/sbin/update &

# Die Tastaturtabelle wird geladen.
/sbin/loadkeys /etc/keytables/de-latin1.map

# Schliesslich wird noch das lokale Initialisierungscript aufgerufen.
/bin/sh /etc/rc.local
```

Das Beispiel zeigt nur ein paar wichtige Kommandos, die typischerweise nach jedem Systemstart ausgeführt werden. Wie in der letzten Zeile gezeigt, können weitere Initialisierungsscripts aus der `/etc/rc` heraus aufgerufen werden. Charakteristisch für den Ablauf der Systeminitialisierung mit `simpleinit` ist der lineare Ablauf: die Scriptdateien werden jedesmal beim Start des Mehrbenutzermodus von Anfang bis Ende durchgearbeitet.

Beim `sysvinit` gibt es wegen der Unterstützung mehrerer Runlevel wesentlich mehr Möglichkeiten für die Systeminitialisierung. Jeder Runlevel zeichnet sich durch eine bestimmte Laufzeitumgebung aus. Beim Wechsel zwischen den Levels müssen einige Komponenten neu hinzugefügt, andere wieder

aus der Umgebung gelöscht werden.

Die Programme oder Shellscripate, die dieses vielschichtige System zur Initialisierung der Runlevel realisieren, werden von der `inittab` aus aufgerufen, wie es in dem Beispiel oben dargestellt wurde.

Bei den Aktionen `sysinit` und `boot` werden einfache Scriptdateien ausgeführt, die eine gemeinsame Systemgrundlage für alle Multiuser-Runlevel schaffen. Diese Shellscripate unterscheiden sich im Prinzip nicht von der oben vorgestellten Datei `/etc/rc` für `simpleinit`.

Die weitere Initialisierung der einzelnen Runlevel erfordert mehr Flexibilität. Die mit den Sourcen vom `sysvinit` ausgelieferten Initialisierungsscripate für die Multiuserlevel arbeiten mit einem Satz hochspezialisierter kleiner Shellscripate, die jeweils für das Hinzufügen oder Löschen einer einzelnen Komponente der Laufzeitumgebung verantwortlich sind. Diese Miniscripate werden nach Bedarf beim Wechsel der Runlevel vom Shellscripate `/etc/rc.d/rc` aus mit den Argumenten `start` oder `stop` aufgerufen.

Für jeden Runlevel wird eine bestimmte Teilmenge der Miniscripate zum Starten einer Komponente genutzt, alle anderen werden mit dem Argument `stop` ausgeführt. Auf diese Weise hat jeder Runlevel seine charakteristische Ausstattung, egal von welchem Level aus dorthin gewechselt wurde.

Um die Komponenten den Runlevels zuzuordnen, benutzt das Shellscripate `/etc/rc.d/rc` von `sysvinit` eine Verzeichnishierarchie unter `/etc/rc.d` mit einem eigenen Unterverzeichnis für jeden Runlevel. Zu Runlevel 1 gehört das Verzeichnis `/etc/rc.d/rc1.d`, zu Level 2 das Verzeichnis `/etc/rc.d/rc2.d` und so weiter.

Ein rekursives Listing von `/etc/rc.d` kann beispielsweise so aussehen:

```
[root@atlantis /root]# ls -RF /etc/rc.d/
init.d/          rc.local*      rc.sysinit*   rc1.d/
rc*              rc.modules*   rc0.d/        rc2.d/

/etc/rc.d/init.d:
funktionen      rc.ibcs*      rc.local*     rc.nfsfs*
rc.cfsfs*      rc.inet*      rc.lpd*       rc.pcmcia*
rc.cron*       rc.isdn*      rc.network*   rc.sendmail*
rc.down*       rc.keytable*  rc.nfs*       rc.syslog*

/etc/rc.d/rc0.d:
K02local@      K30nfsfs@     K48isdn@     K70syslog@    K90network@
K10lpd@        K32cfsfs@     K50inet@     K86pcmcia@    S00down@
K20sendmail@   K40nfs@       K60cron@     K88ibcs@

/etc/rc.d/rc1.d:
K30nfsfs@      S10network@   S30syslog@   S54isdn@      S96keytable@
K32cfsfs@      S20pcmcia@    S40cron@     S80sendmail@  S98local@
K40nfs@        S22ibcs@     S50inet@     S90lpd@

/etc/rc.d/rc2.d:
S10network@    S30syslog@    S54isdn@     S72cfsfs@     S96keytable@
S20pcmcia@     S40cron@      S60nfs@      S80sendmail@   S98local@
S22ibcs@       S50inet@     S70nfsfs@    S90lpd@
```

```
[root@atlantis /root]# _
```

Beim Wechsel in einen von dem rc-Script kontrollierten Runlevel werden die Miniscripts aus dem korrespondierenden Verzeichnis aufgerufen. Die Miniscripts, deren Dateinamen mit K beginnt, werden mit dem Argument stop aufgerufen, die mit Dateinamen auf S mit dem Argument start.

Wegen möglicher Abhängigkeiten der Miniscripts voneinander ist die Reihenfolge ihrer Bearbeitung von Bedeutung. Deshalb wird durch die zweistellige Zahl im Dateinamen die Reihenfolge ihrer Verarbeitung bestimmt.

Weil in jedem der Runlevel-Verzeichnisse die gleichen Miniscripts nur unter anderen Namen aufgerufen werden, sind die Dateinamen nur symbolische Links auf die echten Dateien in dem Verzeichnis /etc/rc.d/init.d. Auf diese Weise müssen nicht mehrere Kopien der gleichen Datei verwaltet werden.

```
#!/bin/sh
# Beispiel eines Initialisierungsscripts fuer den Druckerdaemon.

# Zuerst wird ein weiteres Shellscript eingefuegt, in dem die
# Hilfsfunktionen starte() und beende() definiert sind:
source /etc/rc.d/init.d/funktionen

# Hier wird getestet, ob die erforderlichen Komponenten existieren:
[ -f /usr/sbin/lpd ] || exit 0
[ -f /etc/printcap ] || exit 0

# Schliesslich werden die Komponenten hinzugefuegt oder geloescht:
case "$1" in
    start)
        # Start des Druckerdaemons
        starte lpd
        touch /var/lock/subsys/lpd
        ;;
    stop)
        # Stop des Druckerdaemons
        beende lpd
        rm -f /var/lock/subsys/lpd
        ;;
    *)
        echo "Usage: rc.lpd {start|stop}"
        exit 1
esac
exit 0
```

Die zu Beginn des Beispiels eingefügte Datei enthält die Definitionen von Hilfsfunktionen, die von allen Initscripts benutzt werden können. Die beiden im Beispiel verwendeten Funktionen starte() und beende() sehen folgendermaßen aus:

```
# Definition von Funktionen fuer die Initscripts.
export PATH="/sbin:/usr/sbin:/bin:/usr/bin"
```

```
# Die Funktion starte() sorgt dafür, daß kein Aufruf doppelt erfolgt.
starte() {
    # Hier wird getestet, ob das Programm bereits laeuft:
    [ "`pidof ${1##*/}`" != "" ] && return

    # Wenn das nicht der Fall ist, wird der Name ausgegeben...
    echo -n "$1 "
    # ...und das Programm gestartet:
    $*
}

# Die Funktion beende() hält einen Systemservice an.
beende() {
    # Zuerst wird die Prozess-ID der Komponente gesucht:
    pid=`pidof ${1##*/}`

    # Wenn die Komponente laeuft, wird sie beendet:
    if [ "$pid" != "" ] ; then
        echo -n "$1 "
        kill -9 $pid
    fi
}
```

Sie finden die Erklärung aller in dem Beispiel verwendeten Symbole und Funktionen zur Shellprogrammierung in dem Abschnitt über die [bash](#).

Das zusammen mit dem `sysvinit` installierte Programm `runlevel` gibt den letzten und den aktuellen Runlevel aus. Diese Information kann in den Initialisierungsscripts ausgewertet werden.

## Auswahl und Änderung des Runlevel

Beim `simpleinit` gibt es nur zwei Runlevel:

- Das System startet im Einbenutzermodus, wenn beim LILO Bootprompt das Wort `single` angegeben wurde oder wenn die Datei `/etc/singleboot` existiert. Sobald die Shell im Single-User-Mode beendet wird, fährt das System weiter in den Mehrbenutzermodus hoch.
- Wenn keine der oben genannten Bedingungen zutrifft, wird das System sofort im Mehrbenutzermodus gestartet.

Beim `sysvinit` wird der erste Runlevel nach dem Booten normalerweise als `initdefault` in der `inittab` festgelegt. Diese Einstellung kann durch die Angabe eines gültigen Runlevel (1 bis 9, S oder `single`) auf dem LILO Bootprompt übergangen werden. Außerdem ist im Notfall auch das Booten einer einfachen Shell vor der Ausführung irgendeines Initialisierungsprogramms möglich, indem auf dem Bootprompt die Option `-b` oder das Schlüsselwort `emergency` angegeben wird. Wenn `init` keine Angabe eines gültigen Runlevel finden kann, wird der Bootprozeß angehalten und auf die Eingabe des Runlevel von der Tastatur gewartet.

Wenn das System läuft, kann der Runlevel auf verschiedene Weisen beeinflußt werden. Eine direkte Veränderung kann durch das Programm `telinit` herbeigeführt werden. `telinit` muß mit

Rootprivilegien aufgerufen werden und erwartet den Runlevel, in den gewechselt werden soll, als Argument auf der Kommandozeile.

Neben den im Abschnitt über die `inittab` bereits beschriebenen Möglichkeiten, den Runlevel durch Signale vom Powermanagement oder von der Tastatur zu verändern, bietet `sysvinit` noch zwei allgemeinere Mechanismen zur Kommunikation mit anderen Prozessen an.

- Wenn der `init`-Prozeß ein `SIGHUP` oder `SIGTERM` erhält, liest er die `inittab` neu ein und prüft, ob eine Datei `/etc/initrundl` existiert. Existiert die Datei und enthält die Bezeichnung eines Runlevel, wird in diesen Level gewechselt.
- `sysvinit` bietet allen Prozessen des Systems die Pipeline `/dev/initctl` zur Übermittlung von Requests an. Durch korrekt formulierte Anfragen in diesem Kanal kann auch der Runlevel geändert werden. Sie finden Informationen über die genaue Verwendung der Pipeline in den Quellen zu `sysvinit`.

## Das System herunterfahren

Bei einem Mehrbenutzersystem sind vor dem Abschalten des Rechners noch ein paar Schritte notwendig, die bei weniger leistungsfähigen Betriebssystemen nicht erforderlich sind.

Bei einem vernetzten Rechner, der möglicherweise von mehreren natürlichen Personen gleichzeitig benutzt wird, muß vor dem Abschalten selbstverständlich darauf geachtet werden, daß alle Benutzer ihre Arbeit rechtzeitig sichern und ihre Sessions beenden können.

Wegen der im Hintergrund arbeitenden Dämonen besteht aber auch bei einem System, daß nur von einer einzigen Person benutzt wird, die Gefahr, daß ein Dämonprozeß im Moment des Abschaltens gerade dabei ist, in eine Datei zu schreiben. Wenn dieser Schreibvorgang nicht abgeschlossen werden kann, befinden sich die Daten in einem inkonsistenten Zustand, was zu unvorhersehbaren und wahrscheinlich unerwünschten Ergebnissen führen kann.

Weil Linux einen Teil des Arbeitsspeichers als Cache zur Beschleunigung der Festplattenzugriffe nutzt, ist die Gefahr, beim unkontrollierten Ausschalten des Rechners Fehler in den Daten auf der Festplatte zu erzeugen, um so größer.

Um diese Probleme zu vermeiden, muß das Betriebssystem heruntergefahren werden, bevor der Rechner ausgeschaltet werden kann. Im einzelnen werden hierzu alle Benutzerprozesse und Dämonen beendet, der Buffercache mit der Festplatte synchronisiert und schließlich alle Dateisysteme demontiert.

Um der Systemverwalterin das Herunterfahren des Systems zu erleichtern, gibt es das Programm [`shutdown`](#). Es sorgt dafür, daß alle eingeloggten Benutzer von dem bevorstehenden Systemhalt informiert werden, es verhindert neue Logins und es führt schließlich die erforderlichen Aktionen für das geordnete Anhalten des Systems aus.

Das mit dem `simpleinit` installierte `shutdown` führt alle Aktionen zum Herunterfahren des Systems selbst aus. Es sendet allen laufenden Prozessen zuerst ein `SIGTERM` und zwei Sekunden später ein `SIGKILL`, dann wird der Swapbereich deaktiviert und die Dateisysteme abgebaut.

Das `sysvinit` kann wesentliche Teile des Shutdown selbst ausführen. Insbesondere werden bei

geeigneter Konfiguration der Miniscripts durch einen Wechsel in den Runlevel 0 alle Komponenten aus der Laufzeitumgebung entfernt, die im Mehrbenutzermodus gestartet wurden.

Das Programm `shutdown`, das mit dem `sysvinit` installiert wird, übernimmt das Timing und die Koordination beim Herunterfahren des Systems. Es benachrichtigt die aktiven User und übergibt normalerweise die Kontrolle für das eigentliche Shutdown wieder zurück an `init`. Die letzte Phase des Anhaltens wird dann an das Programm `halt` abgegeben, das alle Aktivitäten des Systems beendet und den Warmstart auslöst.

Das `sysvinit` kann ein Shutdown als Reaktion auf die Tastankombination ALT-CTRL-DEL ausführen. Damit nicht jede Person, die Zugang zur Tastatur der Console hat, das System einfach anhalten kann, hat die Systemverwalterin die Möglichkeit, in der Datei `/etc/shutdown.allow` die Namen der User anzugeben, die zum Herunterfahren des Systems berechtigt sind.

Wenn aus irgendwelchen Gründen das System nicht mit `shutdown` angehalten werden kann, ist es auch möglich, das System „per Hand“ herunterzufahren. Voraussetzung dafür ist natürlich, daß Sie Root-Privilegien haben.

Für ein geordnetes Herunterfahren des Rechners müssen alle Benutzerprozesse angehalten werden. Nachdem Sie alle aktiven natürlichen Systembenutzer mit einer `wall`-Message gewarnt haben, können Sie (nach einer angemessenen Zeit) die Dämonen, Anwenderprogramme und Loginshells durch die Programme `kill` und `killall` mit `SIGTERM`, später mit `SIGKILL` beenden. Dabei müssen Sie darauf achten, daß Sie nicht aus Versehen auch die Shell beenden, mit der Sie gerade arbeiten. Um das Einloggen weiterer Systembenutzer zu verhindern, können Sie die Datei `/etc/nologin` anlegen. Außerdem kann `init` veranlaßt werden, keine neuen Prozesse zu starten, indem ihm mit dem Programm `kill` das Signal `SIGSTP` gesendet wird.

Wenn alle Prozesse außer `init` und der letzten Shell beendet sind, können die Dateisysteme mit dem Kommando `umount -a` demontiert werden. Schließlich wird das Schreibrecht für das Rootfilesystem durch den Befehl `mount -n -o ro,remount /` weggenommen. In diesem Zustand ist das System vollständig heruntergefahren und der Rechner kann gefahrlos abgeschaltet werden.

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [Laufzeitmodule für den Kernel](#) **Up:** [Systemverwaltung](#) **Previous:** [Systemverwaltung](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

## Subsections

- [Erzeugung und Installation von Kernelmodulen](#)
    - [Zusammenhang zwischen Modulen und Kernelversion](#)
    - [Aufbewahrung der Kernelmodule](#)
  - [Laden und Entfernen von Kernelmodulen](#)
    - [Die Konfigurationsdatei `/etc/conf.modules`](#)
    - [Initialisierungsparameter für modulare Gerätetreiber](#)
    - [Entfernen von Kernelmodulen](#)
    - [Automatisches Laden und Entfernen von Modulen durch den Kerneldämon](#)
- 

# Laufzeitmodule für den Kernel

Linux erlaubt das Hinzufügen modularer Gerätetreiber oder Funktionen in den laufenden Kernel. Ursprünglich wurde die Modulschnittstelle für den Linux-Kernel entwickelt, um den Programmierern bessere Möglichkeiten zum Testen neuer Gerätetreiber zu bieten. Es hat sich aber schnell gezeigt, daß die Modularisierung des Kernels viel mehr Vorteile bietet, deshalb wurden nach und nach fast alle Gerätetreiber und viele Kernelfunktionen -- wenigstens potentiell -- zu Modulen umgearbeitet.

Die Distributoren können mit einem vollständig modularisierten Kernel kleinere Installationssysteme herstellen, die zudem besser an das Zielsystem angepaßt sind als die überladenen generischen Kerneldateien auf den Bootdisketten der Vergangenheit.

Die Benutzer eines modularisierten Kernels profitieren von der Schlankheitskur des Betriebssystems. Alle Gerätetreiber, die nicht ständig in Benutzung sind, können als Module ausgelagert und nur bei Bedarf in den Speicher geladen werden.

## Erzeugung und Installation von Kernelmodulen

Um aus einem normalen Treiber ein Modul zu machen, müssen zunächst die Sourcen des Treibers mit dem Compiler übersetzt werden. Zusätzlich ist in der Objektdatei zu jedem Treiber jeweils eine Funktion `init_module` und `cleanup_module` erforderlich, mit der das Modul sich selbst initialisieren und wieder auflösen kann.

Die Kernelquellen enthalten für alle Treiber, deren Modularisierung sinnvoll ist, die entsprechenden Zusatzfunktionen und können deshalb sowohl als fester Kernelbestandteil als auch als Modul übersetzt werden. Das `Kernelmakefile` und das Konfigurationsscript unterstützen die Modularisierung und automatisieren die eigentliche Erzeugung der ladbaren Objektdateien. Die Systemverwalterin muß lediglich beim Konfigurieren eines neuen Kernels vor dem Übersetzen der Sourcen die gewünschten Module auswählen, danach die Module mit dem Kommando ```make modules```



compilieren und schließlich durch das Kommando `make modules_install` in das dafür vorgesehene Verzeichnis kopieren lassen.

Bei der Auswahl der Gerätetreiber, die als Modul übersetzt werden sollen, muß auf eventuelle Abhängigkeiten der Module untereinander geachtet werden. Die kritischen Abhängigkeiten werden vom Konfigurationsscript erkannt und nötigenfalls automatisch korrigiert. Beispielsweise ist es unmöglich, gleichzeitig den Dateisystemtyp `msdos` fest in den Kernel einzubauen und den Basistyp `fat` als Modul davon zu trennen.

## Zusammenhang zwischen Modulen und Kernelversion

Die Kernelmodule sind nicht unabhängig von der Kernelversion, für die sie erzeugt wurden. Jedes Modul benutzt mehr oder weniger viele Funktionen und globale Variable, die vom laufenden Kernel exportiert werden und dem "Kernellinker" `insmod` als *Symbole* zum Linken zur Verfügung stehen. Diese exportierten Symbole bilden die Schnittstelle, an der die Module angekoppelt werden. Eine Liste der Symbole steht in der Pseudodatei `/proc/ksyms` und kann vom Systemprogramm `ksyms` angezeigt werden.

Wenn eine der exportierten Funktionen in einer neuen Kernelversion verändert wird, müssen alle Module, die diese Funktion benutzen, neu übersetzt werden. Aus diesem Grund werden einfacherweise die Kernelmodule für jeden Patchlevel des Kernels neu übersetzt. Das Systemprogramm `insmod` kann die Kernelversionsnummer, für die ein Modul übersetzt wurde, aus der Moduldatei lesen und verweigert normalerweise das Laden eines Moduls, wenn dessen Versionsnummer nicht mit der des laufenden Kernels übereinstimmt.

Dieses einfache Schema der Zuordnung von Modulen zur passenden Kernelversion ist in zweierlei Hinsicht unzureichend. Einerseits können im Zuge eigener Entwicklungsarbeit Veränderungen am Kernel und damit an der Modulschnittstelle stattfinden, ohne daß die Versionsnummer hochgezählt wird. Andererseits ist nicht jedes Modul vom Upgrade des Kernels auf den nächsten Patchlevel betroffen. Es ist durchaus möglich, daß ein Modul ohne Veränderung für viele Kernelversionen benutzt werden kann, wenn sich an den von diesem Modul benutzten Kernelfunktionen nichts ändert.

Um die Abhängigkeit der Module von der Versionsnummer des Kernels zu trennen und gleichzeitig die Verwendung älterer Module mit neuen Kernelversionen zu unterstützen, können Kernel und Module mit speziellen Versionsinformationen für die Modulschnittstelle ausgestattet werden. Um dieses Feature zu nutzen, muß der Kernel mit der Option `CONFIG_MODVERSIONS` übersetzt werden.

Die Namen aller exportierten Kernelsymbole werden dadurch beim Übersetzen des Kernels automatisch um eine Prüfsumme erweitert, die aus dem Programmtext für die entsprechende Funktion generiert wurde. Auf diese Weise kann sichergestellt werden, daß nur solche Module zum Kernel hinzugelinkt werden, die zur aktuellen Schnittstelle passen.

## Aufbewahrung der Kernelmodule

Damit die Module für verschiedene Kernelversionen nicht durcheinander geraten, hat jede Kernelversion ihr eigenes *Modulverzeichnis*. Die Module werden bei der automatischen Installation durch das Kommando `make modules_install` in ein Unterverzeichnis von `/lib/modules` kopiert, dessen Name mit der Versionsnummer des Kernels übereinstimmt, für den sie übersetzt wurden. Wenn Sie beispielsweise die Module für Linux-2.0.18 übersetzt und installiert haben, befinden sich die Objektdateien der Module für diese Kernelversion in dem Verzeichnis `/lib/modules/2.0.18`.

Zusätzlich werden die Module nach den Typen `block`, `cdrom`, `fs`, `ipv4`, `misc`, `net` und `scsi` geordnet und in entsprechenden Unterverzeichnissen aufbewahrt.

Alle Programme, die mit Kernelmodulen umgehen, suchen automatisch in dem zur laufenden Kernelversion gehörenden Modulverzeichnis nach dem Modul, das geladen werden soll.

## Laden und Entfernen von Kernelmodulen

Zum Laden der Laufzeitmodule steht zunächst der Kernellinker `insmod` zur Verfügung, mit dem ein einzelnes Modul in den Kernel eingefügt werden kann. Wie der Linker des GCC versucht `insmod`, die unaufgelösten Symbole des Moduls mit den exportierten Symbolen des Kernels zu verbinden.

Wenn alle ``offenen Enden" des Moduls erfolgreich mit dem Kernel verknüpft worden sind, veranlaßt der Linker die Initialisierung des Moduls durch die Modulfunktion `init_module`. Diese Funktion ist weitgehend mit der Initialisierungsfunktion identisch, die beim Booten eines fest installierten Treibers vom Kernel aufgerufen wird. Einige Treiber erlauben die Übergabe von Initialisierungsparametern, ähnlich der Bootparameter vom LILO Bootprompt.

```
[01] # insmod lp io=0x378 irq=7
[02] # insmod 3c503 io=0x280 xcvr=0
[03] # insmod msdos
fat_add_cluster: wrong version or undefined
fat_statfs: wrong version or undefined
fat_put_super: wrong version or undefined
...
Loading failed! The module symbols don't match your linux-2.0.18
[04] # _
```

Es ist möglich, daß ein Modul von einem anderen Modul abhängig ist. Das ist der Fall, wenn ein Modul Funktionen benutzt, die von einem anderen exportiert werden. In dem Beispiel werden vom Dateisystemtyp `msdos` Funktionen des Basistyps `fat` gebraucht. Um `msdos` als Modul laden zu können, muß deshalb das Modul für `fat` bereits geladen sein.

Diese Abhängigkeitsbeziehung von Modulen untereinander führt zu Ketten oder *Modulstacks*. Um einen solchen Stack mit `insmod` zu laden, müssen alle Module nacheinander in der richtigen Reihenfolge an den Kernel angehängt werden.

Um das Laden von Modulstacks zu erleichtern, gibt es das Programmpaar `depmod` und `modprobe`. Mit `depmod` wird eine Datei erzeugt, in der die Abhängigkeit der Module aufgezeichnet ist. `modprobe` hat die gleiche Aufgabe wie `insmod`, es sorgt aber dafür, daß der Modulstack eines abhängigen Moduls komplett geladen wird. Dazu benutzt es die von `depmod` erzeugte Datei.

```
[01] # uname -sr
Linux 2.0.18
[02] # depmod -a
[03] # depmod -a 2.0.14
*** Unresolved symbols in module /lib/modules/2.0.14/fs/vfat.o
*** Unresolved symbols in module /lib/modules/2.0.14/misc/iBCS
*** Unresolved symbols in module /lib/modules/2.0.14/net/dummy.o
*** Unresolved symbols in module /lib/modules/2.0.14/net/ppp.o
[04] # modprobe msdos
[05] # modprobe -a -t fs \*
[06] #
```

Das zweite Kommando führt zur automatischen Prüfung der Abhängigkeiten aller Module, die im Modulverzeichnis für den laufenden Kernel (2.0.18) enthalten sind. Das dritte Kommando testet alle Module im Verzeichnis für die Kernelversion 2.0.14 auf Abhängigkeiten. Die Warnungen zeigen, daß einige dieser Module nicht zur laufenden Kernelversion passen. Die Abhängigkeit der Module untereinander wird trotzdem korrekt erkannt.

Das vierte Kommando zeigt den Vorteil von `modprobe` gegenüber `insmod`: beim Laden eines abhängigen Moduls wird automatisch der gesamte Modulstack geladen. Im fünften Kommando des Beispiels wird schließlich gezeigt, wie mit `modprobe` alle Module des Typs `fs`, also die Dateisysteme, automatisch geladen werden können. Ohne die Angabe eines Typs würde `modprobe` alle Module für den laufenden Kernel laden.

## Die Konfigurationsdatei `/etc/conf.modules`

Das Programm `modprobe` kann durch die Konfigurationsdatei `/etc/conf.modules` auf eine besondere Systemumgebung und für spezielle Aufgaben eingestellt werden. Für alle möglichen Einstellungen benutzt das Programm eine lange Liste von Vorgaben, die mit der Kommandozeilenoption `-c` angezeigt wird. Die Liste läßt sich durch zusätzliche Einträge in der Konfigurationsdatei ergänzen, und die Vorgaben können durch eigene Definitionen ersetzt werden.

```
# Beispiel fuer /etc/conf.modules
keep
path[boot]=/usr/lib/modules

alias scsi_hostadapter aha1542

options lp io=0x378 irq=7

post-install lp tunelp /dev/lp1 -i 7
```

In den ersten Zeilen des Beispiels wird der Suchpfad erweitert, auf dem `modprobe` nach den angeforderten Kernelmodulen sucht. Der vorgegebene Suchpfad beginnt in `/lib/modules/boot`, geht dann über die Unterverzeichnisse des Modulverzeichnisses der laufenden Kernelversion und danach durch die Unterverzeichnisse von `/lib/modules/default`. Ohne das Schlüsselwort `keep` wird der vorgegebene Suchpfad ersetzt und nicht, wie in diesem Fall, erweitert.


Die Definition von Aliasnamen hat für die direkte Verwendung mit `modprobe` nur kosmetische Bedeutung. In Kombination mit dem Kerneldämon `kerneld` wird über Aliasnamen die Auswahl eines bestimmten Moduls für eine allgemeinere Geräteklasse vorgenommen, wie in dem Beispiel für einen SCSI-Hostadapter gezeigt.

Beim Laden eines einzelnen Moduls kann `modprobe`, wie `insmod`, Symboldefinitionen als Argumente für die Modulinitialisierung übernehmen. Beim Laden eines Modulstacks oder einer Modulgruppe ist das nicht möglich, weil sich die Kommandozeilenargumente nicht den einzelnen Modulen zuordnen lassen. Stattdessen können in der Konfigurationsdatei für jedes Modul Optionen festgelegt werden, mit denen der Treiber beim Laden eingestellt wird.

Für aufwendigere Modulinitialisierungen besteht die Möglichkeit, vor und nach dem Laden des Moduls beliebige Kommandos ausführen zu lassen.

# Initialisierungsparameter für modulare Gerätetreiber

Die Definition von Symbolen beim Laden eines Moduls durch `insmod`, beziehungsweise eines der Front-Ends `modprobe` oder `kerneld`, hat die Rolle des Bootparameters, der dem Kernel zur Initialisierung eines Gerätetreibers mit dem Bootprompt übergeben werden kann. Mit Hilfe dieser Symboldefinitionen können IO-Adressen, Interruptnummern und sonstige Angaben über die installierte Hardware an den modularen Treiber übergeben werden.

Leider werden zur Übergabe der Parameter in den beiden Anwendungsfällen ganz verschiedene Methoden benutzt.  Deshalb können die meisten modularen Treiber nicht einfach mit dem Bootparameter vom LILO Bootprompt initialisiert werden. Die Tabelle [5.3](#) zeigt die Symbole zur Initialisierung der wichtigsten modularen Treiber.

Gerät & Modul & Symbole  
Die Symbole zur Modulinitialisierung

Aztech CD	aztcd	<b>azt_port=IO-Port</b>
Sony CDU31a	cdu31a	cdu31a_port=IO-Port cdu31a_irq=IRQ sony_pas_init=FLAG
Phillips CM206	cm206	cm206=IO-Port, IRQ
GoldStar	gscd	gscd=IO-Port
Mitsumi MultiSession	mcdx	mcdx_drive_map=IO-Port, IRQ[, IO-Port, IRQ]
Optics Storage	optcd	optcd_port=IO-Port
CreativeLabs	sbpcd	sbpcd=IO-Port, Typ
Sanyo CDR-H94A	sjcd	sjcd_base=IO-Port
Sony CDU535	sonycd535	sony535_cd_base_io=IO-Port sony535_irq_used=IRQ
Ethernet	./net	io=IO-Port irq=IRQ
NCR 53c7,8x	53c7,8xx	base=Memory io_port=IO-Port irq=IRQ dma=DMA perm_options=Option
AM53/79C974 PCI	AM53C974	host_scsi_id=ID target_scsi_id=ID max_rate=Rate max_offset=Offset
BusLogic	BusLogic	IO_Address=IO-Port
NCR 53c406a	NCR53c406a	port_base=IO-Port irq_level=IRQ fast_pio=XX dma_chan=XX bios_base=XX
Adaptec AHA152X	aha152x	aha152x=IO, IRQ, ID, Recon, Par...

Adaptec AHA1740	aha1740	slot=XX base=IO-Port irq_level=IRQ
Adaptec AHA-2940	aic7xxx	aic7xxx_extended=1 aic7xxx_no_reset=1 aic7xxx_irq_trigger={-1,0,1}
Future Domain 16xx	fdomain	port_base=XX bios_base=XX interrupt_level=XX this_id=XX
Always IN2000	in2000	setup_strings=Liste

Sie können mit `insmod` alle exportierten statischen Variablen initialisieren. Arrays können durch eine kommaseparierte Liste von Werten belegt werden. Sie finden die in Frage kommenden Symbole durch eine Untersuchung der Moduldatei mit Hilfe des Tools `nm`. In der Tabelle erkennen Sie die Symbole am Buchstaben `d`.

## Entfernen von Kernelmodulen

Einer der großen Vorteile von Kernelmodulen ist die Möglichkeit, sie wieder aus dem Kernel zu entfernen, sobald sie nicht mehr gebraucht werden. Der für diesen Zweck vorgesehene Antagonist von `insmod` ist das Kommando `rmmod`. Mit `rmmod` lassen sich auch ganze Modulstacks entfernen.

Bevor ein Modul aus dem Kernel herausgelöst und sein Speicherbereich freigegeben werden kann, muß es vollständig inaktiv sein. Es müssen alle Prozesse beendet sein, die das Modul benutzt haben. Auf den modularen Treiber einer Ethernetkarte dürfen auch keine Routen mehr laufen. Ist das Modul noch aktiv, während es mit `rmmod` entfernt werden soll, wird eine Fehlermeldung ausgegeben und das Modul unverändert im Kernel belassen.

Das Programm `modprobe` kann mit dem Schalter `-r` ebenfalls zum Löschen von Modulen eingesetzt werden. Es arbeitet dann wie `rmmod`.

## Automatisches Laden und Entfernen von Modulen durch den Kerneldämon

Im letzten Schritt zum perfekt modularisierten Kernel werden die Module bei Bedarf automatisch geladen und selbständig wieder entfernt, sobald sie nicht mehr benötigt werden. Diese Funktionalität wird durch das Zusammenspiel des Kernels mit einem speziellen Kerneldämon im Userspace realisiert. Zur Kommunikation mit dem `kerneld` benutzt der Kernel die Funktionen der Interprozeßkommunikation, *IPC*. Voraussetzungen zur Benutzung dieses Service sind dessen Aktivierung bei der Kernelkonfiguration vor dem Übersetzen der Sourcen (`CONFIG_KERNELD`) und ein im Hintergrund laufender Kerneldämon.

Bei jedem Versuch, auf ein unbekanntes Gerät, ein neues Binärformat, ein fremdes Netzwerkprotokoll oder ein bisher nicht unterstütztes Dateisystem zuzugreifen, übermittelt der Kernel eine Aufforderung zum Laden des fehlenden Moduls an den Dämon. Der Dämon gibt den Auftrag an das Programm `modprobe` weiter, das seinerseits versucht, das passende Modul zu finden und in den Kernel einzufügen.

Wenn das nötige Modul nicht geladen werden konnte, gibt der Kernel die übliche Fehlermeldung `"No such device"` aus, sonst arbeitet er ganz normal mit dem Modul weiter.

In der Aufforderung zum Laden eines Moduls benutzt der Kernel in der Regel nicht den Namen des Moduls, sondern eine Bezeichnung der fehlenden Funktion. Wenn zum Beispiel der Floppytreiber als Modul aus dem Kernel ausgelagert wurde und ein Prozeß versucht, auf ein Diskettenlaufwerk zuzugreifen, schickt der Kernel dem `kernel` eine Anfrage der Form: `request_module ('block-major-2')`. Hinter `block-major-2` verbirgt sich der Floppytreiber, das Blockdevice mit Hauptgerätenummer 2. Die Übersetzung in dieser Bezeichnung in den Namen des Kernelmoduls wird von `modprobe` durch einen Aliasnamen vorgenommen. In der Liste von Vorgaben, die in `modprobe` ohne weiteres Zutun enthalten ist, sind die Aliasnamen für alle gängigen Kernelmodule enthalten.

Es gibt jedoch einige Module, die nicht anhand der vom Kernel abgegebenen Anfrage identifiziert werden können. Wenn zum Beispiel auf ein SCSI-Bandlaufwerk zugegriffen werden soll und außer dem Treiber für das Bandgerät auch der Treiber für den SCSI-Hostadapter als Modul geladen werden muß, fordert der Kernel nur ein Modul mit der Bezeichnung `scsi_hostadapter`. Der Kernel hat keine Möglichkeit festzustellen, welcher Hostadapter in den Rechner eingebaut ist.

In diesem Fall muß das passende Modul durch eine entsprechende Aliasdefinition in der Datei `/etc/conf.modules` identifiziert werden.

Kernelmodule, die mit dem `kernel` geladen wurden, sind bei einem Listing der aktiven Module durch `lsmod` mit dem Zusatz `(autoclean)` gekennzeichnet. Das bedeutet, daß diese Module automatisch entfernt werden, wenn sie länger als eine Minute nicht mehr benutzt werden.

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [Prozeßordnung](#) **Up:** [Systemverwaltung](#) **Previous:** [Der Anfang und](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

## Subsections

- [Entstehung der Prozesse: fork und exec](#)
    - [Copy on Write und Demand Loading](#)
  - [Prozeßgruppen, Sessions und kontrollierende Terminals](#)
  - [Prozeßtabelle und Programmumgebung](#)
  - [Abstürzende Programme und hängende Prozesse](#)
    - [Prozesse durch Signale beenden](#)
    - [Zombies und blockierte Prozesse](#)
  - [Systemabsturz](#)
- 

# Prozeßordnung

Ein lauffähiges Programm, das sich im Arbeitsspeicher unter der Kontrolle des Betriebssystems befindet, wird als *Prozeß* bezeichnet. Zu einem Prozeß gehören mehr als nur die binären Daten aus der Programmdatei, die vom Prozessor abgearbeitet werden. Zu einem Prozeß gehören unter anderem auch

- der virtuelle Adressraum, in dem das Programm arbeitet,
- Dateien und Gerätedateien, die vom Programm geöffnet wurden,
- die Prozeßumgebung und weitere Datenstrukturen, die der Kernel für den Prozeß verwaltet.

## Entstehung der Prozesse: fork und exec

Prozesse entstehen nicht spontan, sondern sie werden von bereits existierenden Prozessen erzeugt. Dazu muß ein laufender Prozeß die beiden Systemaufrufe `fork` und `exec` benutzen. Nach einem `fork` erzeugt der Kernel eine genaue Kopie des laufenden Prozesses. Die Kopie erhält einen neuen Eintrag in der Prozeßtabelle mit einer neuen Prozeßnummer und einen eigenen virtuellen Adressraum. Der erzeugende Prozeß wird als Elternprozeß bezeichnet, der neue Zweig als das Kind. Als Kopie seines Erzeugers ``erbt" ein Kindprozeß die Prozeßumgebung und die offenen Dateien von seinem Elternprozeß.

Eltern und Kind arbeiten unabhängig voneinander weiter. Zunächst ist der Programmtext der beiden Zweige identisch, der Programmablauf wird aber in der Regel nach dem `fork` unterschiedlich weitergeführt. Durch den Systemcall `exec` kann das Kind nun den Kernel veranlassen, eine neue Programmdatei von der Festplatte zu lesen und damit den virtuellen Speicherbereich des Kindes zu überschreiben. Auf diese Weise kommt das neue Programm zur Ausführung.

Bei der Verdrängung des Programmtextes im virtuellen Speicherbereich des Kindes bleibt die Prozeßnummer und die Prozeßumgebung erhalten. Auch das Verwandtschaftsverhältnis zum Elternprozeß mit allen daraus resultierenden Rechten bleibt bestehen.

## Copy on Write und Demand Loading

Wenn der Kernel nach dem Systemcall `fork` einen Prozeß kopiert, wird der Speicherinhalt nicht wirklich dupliziert. Die virtuelle Speicherverwaltung des Kernels benutzt die gleichen physikalischen Speicherbereiche für Eltern und Kind, solange keiner der beiden Prozesse deren Inhalt verändert hat. Erst wenn in eine Speicherseite geschrieben wird, erzeugt der Kernel die Kopie und tauscht die veränderte Seite in dem virtuellen Arbeitsspeicher des schreibenden Prozesses aus. Diese sehr schnelle und sparsame Art der Speicherverwaltung wird als *copy on write* bezeichnet.

Beim Laden neuer Programmdateien von der Festplatte geht das Betriebssystem ähnlich sparsam vor, indem es nur die Teile des Programmtexts in den Arbeitsspeicher liest, die unmittelbar für den Programmablauf gebraucht werden. Dieses *demand loading* verkürzt die Zeit von der Eingabe eines Kommandos bis zu seiner Ausführung erheblich.

## Prozeßgruppen, Sessions und kontrollierende Terminals

Nicht jedes Kommando, das Sie der Shell zur Bearbeitung übergeben, besteht aus einem einfachen Programmaufruf. Bei einer Pipeline müssen mehrere Programme geladen und bei der Ausführung miteinander verbunden werden. Die Shell erzeugt für jedes Programm mit der oben beschriebenen Methode durch `fork` und `exec` einen eigenen Prozeß. Um die Prozesse der Pipeline gemeinsam kontrollieren zu können, faßt die Shell die Prozesse in einer *Prozeßgruppe* zusammen. Die ID der Prozeßgruppe ist mit der Prozeß-ID des ersten Prozesses der Pipeline identisch, dieser Prozeß ist der Führer der Gruppe.

Eine *Session* umfasst eine oder mehr Prozeßgruppen. Für jede Loginshell wird vom Programm `login` automatisch eine neue Session eröffnet. Die Session trägt die gleiche ID wie der Prozeß der Loginshell. Die Loginshell ist damit Führer einer Session.

Die Eröffnung von Sessions und Prozeßgruppen ist nicht auf die angeführten Beispiele mit den Shells beschränkt. Jeder neue Prozeß kann eine neue Prozeßgruppe oder Session erzeugen.

Solange ein Prozeß keine neue Session eröffnet, ist er in der gleichen Session wie sein Elternprozeß. Der führende Prozeß einer Session kann mit einem Terminal verbunden sein, das dann als das *kontrollierende Terminal* bezeichnet wird. Jedes Terminalgerät kann nur eine Session kontrollieren. Von den Prozeßgruppen einer Session kann nur eine mit dem kontrollierenden Terminal verbunden sein. Diese Prozeßgruppe arbeitet im Vordergrund, alle andern Prozeßgruppen arbeiten im Hintergrund. Die Prozesse können nur auf das kontrollierende Terminal ihrer Session zugreifen.

## Prozeßtabelle und Programmumgebung


Für jeden Prozeß verwaltet Linux das umfangreiche *Task-Struct*, eine Datenstruktur, in der alle wichtigen Informationen über den Prozeß verzeichnet sind. Ein großer Teil dieser Daten wird von dem Programm `ps` aufbereitet und übersichtlich angezeigt.



Unter anderem sind dort die Prozeßnummer des Prozesses selbst und seiner Eltern, die realen und effektiven IDs von Eigentümer und Benutzergruppe des Prozesses, die Ressourcen-Limits und die verbrauchten Ressourcen verzeichnet.

# Abstürzende Programme und hängende Prozesse

Es gibt kein fehlerfreies Programm und überall Benutzer, die einem sonst sehr zuverlässigen Programm ganz erstaunliches und unerklärliches Verhalten entlocken, und es gibt das Naturgesetz von Murphy ...

Deshalb kommt es mit Sicherheit bei jedem Rechner irgendwann einmal zu einem Programmabsturz. Weil aber jedes Programm in einem eigenen, vom übrigen System hermetisch abgeschirmten virtuellen Adressraum arbeitet, hat ein Programmabsturz keine unkontrollierbaren Auswirkungen auf das gesamte System.  Beim Versuch, auf den Speicherbereich eines anderen Programms zuzugreifen, wird jeder Prozeß sofort mit dem *SIGSEGV* Signal abgebrochen. Der Prozeß wird dadurch aus dem Arbeitsspeicher gelöscht und kann keinen Schaden mehr anrichten.

Die einzige wirklich kritische Situation kann entstehen, wenn ein Prozeß absichtlich oder aus Versehen den gesamten Arbeitsspeicher verbraucht. Dieser Gefahr kann durch Beschränkung der Ressourcen für User und Prozesse durch [ulimit](#) begegnet werden.

Es kann auch passieren, daß sich ein Programm ``aufhängt''. So ein Prozeß stürzt nicht wirklich ab und bleibt deshalb lauffähig im Arbeitsspeicher. Wegen eines internen Fehlers, beispielsweise einer Endlosschleife, hat sich das Programm jedoch in einem sinnlosen Zweig des Programmablaufs verfangen und kommt nicht zu seinem vorgesehenen Abschluß.

In solch einem Fall kann ein Prozeß meistens durch ein Unterbrechungssignal von der Tastatur angehalten werden. Durch die Tastenkombination CONTROL-C (^C) wird der mit dem kontrollierenden Terminal verbundenen Vordergrundprozeßgruppe das Signal SIGINT geschickt. Durch die Tastenkombination CONTROL-Z (^Z) erhält die gleiche Gruppe das Signal SIGSTP.

## Prozesse durch Signale beenden

Wenn ein Prozeß die Verbindung zu einem kontrollierenden Terminal aufgegeben oder verloren hat, kann er durch die Tastatursignale nicht mehr beeinflußt werden. Wenn nicht durch die Begrenzung einer Ressource, beispielsweise der CPU-Zeit, der Prozeß vom Kernel suspendiert wird, läßt sich das Programm nur noch durch ein extern generiertes Signal beenden.

Das Programm `kill` generiert solch ein Signal und sendet es an den gewünschten Prozeß. Der Kernel leitet das Signal nur weiter, wenn es von einem berechtigten User abgeschickt wurde. Zum Senden von Signalen sind außer der Systemverwalterin mit Rootprivilegien der reale und der effektive Eigentümer des Prozesses befugt.

Eine Liste aller Signale, die mit `kill` generiert werden können, erhalten Sie durch das Kommando ```kill -l`'' (Option *list*). Die Signale, die normalerweise zur Beendigung eines Prozesses führen, sind SIGTERM (15), SIGHUP (1) und SIGINT (2). Diese Signale können von dem Prozeß, der sie empfängt, abgefangen und in einer Fehlerroutine behandelt werden. Auf diese Weise kann beispielsweise ein Editor die geöffneten Dateien noch ordnungsgemäß schließen, bevor er als

Reaktion auf das Signal terminiert. Das Signal `SIGKILL` kann nicht abgefangen werden und führt zur sofortigen Beendigung eines Prozesses.

Das `kill`-Kommando benötigt auf der Kommandozeile eine Identifikation des Prozesses, an den das Signal geschickt werden soll. Dazu kann jedes `kill`-Kommando die Prozeßnummer dieses Prozesses verarbeiten. Um diese Prozeßnummer (`PID`) herauszubekommen, kann das `ps`-Kommando benutzt werden. Mit der Option `-ax` zeigt es alle Prozesse mit ihren Prozeßnummern, den Terminals (falls sie noch kontrollierende Terminals haben), dem Status und dem Namen an.

## Zombies und blockierte Prozesse

Gelegentlich werden in Ausgabe von `ps` Prozesse angezeigt, die mit dem Status `Z` als *Zombie* gekennzeichnet sind. Diese "lebendigen Toten" existieren normalerweise nur für einige Augenblicke. Sie entstehen, wenn ein Prozeß beendet ist, und sie verschwinden, sobald der Elternprozeß das Signal von der Beendigung seines Kindes erhalten und bestätigt hat.

Wenn ein Zombie nicht aus der Prozeßtabelle verschwindet, bedeutet das, daß der Elternprozeß des Zombies eigentlich auf das Signal von der Beendigung seines Kindes warten wollte, jedoch aus irgendwelchen Gründen nicht mehr existiert.

Ein Zombie kann auch durch ein `SIGKILL` nicht aus der Prozeßtabelle entfernt werden. Weil der eigentliche Prozeß nicht mehr existiert und weder Arbeitsspeicher noch Rechenzeit verbraucht, hat ein Zombie außer dem unschönen Eintrag in der Anzeige von `ps` keine nachteilige Auswirkung auf das laufende System.

Es gibt noch weitere Fälle, in denen ein Prozeß auch durch das Signal `SIGKILL` nicht sofort beendet werden kann. Die Ursache hierfür liegt meistens in einem blockierten Systemaufruf. Diese Situation entsteht beispielsweise, wenn ein Prozeß auf die Beendigung einer Schreib- oder Leseoperation eines langsamen Gerätes wartet.

## Systemabsturz

Was für die Anwenderprogramme gilt, ist auch für den Kernel selbst nicht verkehrt. Auch wenn der Linux-Kernel eine beachtliche Stabilität erreicht hat und auf vielen Systemen unter hoher Last wochenlang ohne Unterbrechung läuft, kann das Betriebssystem nicht auf alle möglichen Ausnahmen vorbereitet sein, und es enthält auch Fehler. Ein Fehler des Betriebssystems ist nicht so leicht abzufangen wie der eines Anwenderprogramms. In einem solchen Fall kommt es häufiger zu einem Systemabsturz; manchmal in Form eines „Kernel Panic“ Systemhalts, manchmal zu einem Reset, manchmal zu einem kompletten Systemstillstand.

Wenn sich ein Systemabsturz irgendwie ankündigt, indem beispielsweise das System immer langsamer wird, obwohl kein rechenzeitintensives Programm läuft, können Sie versuchen, den Rechner mit einem `halt`-Kommando oder der Tastenkombination `ALT-CONTROL-DELETE` anzuhalten und so den Schaden zu begrenzen. In jedem Fall sollten Sie alle normalen Benutzeraktivitäten am Rechner beenden und häufig das Systemprogramm `sync` ausführen.

Wenn der unerfreuliche Fall eines echten Systemabsturzes eingetreten ist, erscheint manchmal eine Meldung der folgenden Form auf dem Bildschirm:

```
unable to handle kernel paging request at address C0000010
Oops: 0002
EIP: 0010:00118348
EFLAGS: 00000246
eax: ffffffff ebx: 00000000 ecx: 00000216 edx: 000003d5
esi: 00105816 edi: 0016709f ebp: 001756f8
ds: 002b es: 002b fs: 002b gs: 002b
Pid: 00, process nr: 00
89 50 04 c7 03 00 00 00 00 c7
```

Aus der Fehlermeldung in der ersten Zeile kann geschlossen werden, daß eine Kernelfunktion versucht hat, auf die Speicheradresse 0x010 zuzugreifen. Weil der Kernel in den logischen Speicherbereich über ein Gigabyte (0xC0000000) verschoben ist, erscheint die Adresse mit diesem Offset.

In der zweiten Zeile erscheinen eine weitere Fehlermeldung und ein Fehlercode. Die Oops Meldung aus diesem Beispiel ist typisch für Fehler des Memory-Management. Die Fehlernummer kann weiteren Aufschluß über die Ursache des Absturzes geben; um sie zu entschlüsseln, müssen Sie die Kernelquellen heranziehen.

Der Extended Instruction Pointer EIP zum Zeitpunkt des Fehlers läßt Rückschlüsse auf die Kernelfunktion zu, die für den Absturz verantwortlich ist. Sie können den Namen der Funktion herausfinden, indem Sie die Symboltabelle des Kernels nach der diese Adresse umfassenden Funktion durchsuchen. Das folgende Kommando erledigt diese Aufgabe:

```
# nm /usr/src/linux/tools/zSystem | sort | grep 00118...
00118294 t _try_to_free_page
00118324 T _free_page
0011847c T ___get_free_page
00118638 t _try_to_unuse
0011878c T _sys_swapoff
00118934 T _sys_swapon
00118c64 T _si_swapinfo
00118cf4 T _do_mmap
00118cf4 t ___gnu_compiled_c
00118cf4 t gcc2_compiled.
00118cf4 t mmap.o
00118ec4 T _sys_mmap
00118f50 T _unmap_fixup
# _
```

Aus dieser Liste läßt sich die verantwortliche Funktion, in diesem Beispiel `free_page`,  leicht herausfinden.

Die EFLAGS und die in den darauffolgenden Zeilen aufgelisteten Inhalte der Prozessorregister können im Einzelfall zur genauen Bestimmung der Fehlerursache ebenso herangezogen werden wie die Maschinencode-Sequenz in der letzten Zeile.



## Subsections

- [Linux und Normalzeit](#)
  - [Linux und Zeitzonen](#)
  - [Systemuhr und CMOS-Uhr](#)
  - [Linux und Echtzeit](#)
- 

# Betrachtungen über die Zeit

Die wesentliche Qualität der Zeit, in einer von relativistischen Verzerrungen freien Welt, besteht im unaufhaltsam konstanten Verstreichen; jeder Zeitpunkt ist eindeutig und einmalig.

Um die zeitliche Dimension der realen Welt in der virtuellen Welt von Linux zu modellieren, benutzt das Betriebssystem den Timer-Chip als Uhr. Diese Uhr soll eine möglichst synchrone Zuordnung aller digitalen Ereignisse zu der realen Zeit ihres Auftretens vornehmen.

Das Betriebssystem ordnet jedem Ticken der Systemuhr ein Datum zu, das der Uhrzeit und dem Kalendertag der realen Welt entsprechen soll. Der Timer-Chip zählt mit einer Frequenz von 1,19318MHz, kann also die Zeit mit einer Auflösung im Mikrosekundenbereich messen.

## Linux und Normalzeit

Wenn die Systemuhr Zeitdifferenzen von wenigen Mikrosekunden feststellen kann, drängt sich die Frage nach der Genauigkeit dieser Uhr auf. Die Beantwortung dieser Frage zerfällt in zwei Teile:

1.

Die exakte Frequenz der Systemuhr unterliegt sowohl fertigungsbedingt als auch wegen Temperatur- und anderen Umwelteinflüssen gewissen Schwankungen. Die dadurch entstehende Drift der Systemzeit beträgt einige Sekunden pro Tag. Die Taktrate der Systemuhr läßt sich justieren und die Drift auf Werte unter einer Sekunde pro Tag reduzieren, wenn eine externe Zeitreferenz ausgewertet wird.

2.

Wenn die Synchronisation der Systemzeit mit der Normalzeit durch manuelle Tastatureingabe erfolgt, hängt die Genauigkeit von den motorischen Qualitäten der eingebenden Person ab. Wenn eine externe Zeitreferenz angeschlossen wird, kann die Systemzeit auf einige Millisekunden genau eingestellt werden.

Die NTP-Zeitfunktionen sind im Kernel enthalten, damit kann Linux höchste Ansprüche an Zeitsynchronisation und Genauigkeit erfüllen. Mit dem Systemcall `adjtimex` läßt sich die Systemuhr ``ruckfrei" einstellen. Durch eine vom Kernel emulierte Nachlaufsynchronisation (*Phase Locked Loop*) kann die Geschwindigkeit der Systemuhr automatisch an eine externe Zeitreferenz angepaßt werden.

Eine externe Referenzzeit kann beispielsweise von einer DCF77-Funkuhr über die serielle Schnittstelle in den Rechner eingespeist werden. Wenn die Zeitmarke, die vom Sekundensignal der Funkuhr erzeugt wird, durch eine spezielle Interruptroutine gespeichert und von geeigneter Software ausgewertet wird, läßt sich die Systemzeit mit dem externen Signal in einem Fehlerintervall von einer Millisekunde synchronisieren.


Eine geeignete Funkuhr gibt es für weniger als 80,-DM,  Treiber und zusätzliche Informationen finden Sie in den Unterlagen zu dem Vortrag, den Harald König zu diesem Thema auf dem zweiten Linux-Kongreß gehalten hat (

→ <http://www.lunetix.de/kongress95/slides/koenig/>).

Bei Internet-Systemen kann die externe Referenzzeit aus dem Netz bezogen werden. Das Client-Server-System xntp bietet sehr umfangreiche Funktionalität zu diesem Zweck. Sie finden die aktuelle Version beispielsweise auf <ftp.informatik.hu-berlin.de> im Verzeichnis `/pub/os/linux/sources/kernel/net-source/tools`. Dieses Paket enthält außer den Sourcen noch jede Menge Information zum Thema Zeitsynchronisation von Computern.

## Linux und Zeitzonen

Die Internet-Funktionalität von Linux stellt höhere Anforderungen an die Datumsfunktionen, als das bei einem isolierten Arbeitsplatzrechner der Fall ist. Stellen Sie sich zum Beispiel vor, daß Sie sich von einem Linux-Rechner in Berlin aus mit einem Rechner in Boston, USA, verbinden und von dort eine Datei auf den lokalen Rechner kopieren. In Berlin ist es 10 Uhr vormittags (Sommerzeit), am MIT in Boston ist es 4 Uhr morgens. Sie dürfen erwarten, daß die Datei auf dem Weg von Boston nach Berlin nicht um 6 Stunden gealtert ist.

Um die Zeit auf allen Rechnern im Internet vergleichen zu können, sind alle Systemuhren auf *Universal Time Coordinated* (UTC, ehemals GMT) eingestellt.  Die Umrechnung der Systemzeit in die lokale Zonenzeit wird erst bei der Datumsausgabe durch die Funktionen der C-Bibliothek vorgenommen.

Die Zeitzone des lokalen Systems wird durch die Datei `/usr/lib/zoneinfo/localtime` bestimmt. Diese Datei sollte für alle Systeme mit Mitteleuropäischer Zeit ein symbolischer Link auf die Zeitzonenbeschreibung MET in dem gleichen Verzeichnis sein.

Jeder User kann die Datumsfunktionen auf eine andere Zeitzone umstellen, indem er in der Umgebungsvariablen TZ den Dateinamen der entsprechenden Zonenbeschreibung angibt.


## Systemuhr und CMOS-Uhr

Der Timer-Chip, mit dem die Systemzeit gemessen wird, ist bei abgeschaltetem Rechner außer Betrieb. Damit die Systemzeit beim nächsten Einschalten wieder aktuell ist, haben alle PCs eine batteriegepufferte Uhr, die das Datum und die Uhrzeit auch ohne Netzspannung weiterzählt.

Diese Uhr wird gelegentlich als *Echtzeituhr* bezeichnet. Weil sie im gleichen Baustein wie das CMOS-RAM (Non-Volatile-RAM) untergebracht ist und die Zeit in dessen Speicherbereich schreibt, wird sie in diesem Buch durchgängig als CMOS-Uhr bezeichnet.

Der Kernel liest gleich zu Beginn der zweiten Initialisierungsphase, noch vor der Initialisierung der Geräte, die Zeit aus der CMOS-Uhr und stellt die Systemzeit entsprechend ein. Dabei wartet der Kernel beim Auslesen der Uhr genau den Zeitpunkt ab, an dem das im Sekundentakt stattfindende Update der CMOS-Zeit abgeschlossen ist, also gerade eine neue Sekunde beginnt. Auf diese Weise kann der Kernel die Systemzeit, mit einer Abweichung im Bereich von Mikrosekunden, genau auf die CMOS-Uhr synchronisieren, obwohl diese Uhr die Zeit nur mit einer Auflösung von ganzen Sekunden anzeigen kann.

Diese Methode zum Stellen der Systemzeit führt nur dann zu einer korrekten Einstellung, wenn die CMOS-Uhr auf *Universal Time Coordinated* (oder auch Greenwich Mean Time) eingestellt ist. Wenn Sie die CMOS-Uhr aus irgendwelchen Gründen auf die lokale Zonenzeit eingestellt lassen müssen, können Sie die Systemzeit in einem der Initialisierungsscripts mit dem Programm `clock` erneut stellen und dabei die Zeitzone berücksichtigen.

Wenn Sie Wert auf eine exakte Systemzeit legen, aber über keine externe Zeitreferenz verfügen, sind Sie auf die CMOS-Uhr angewiesen. Da der Hersteller Ihres PC mit Sicherheit keinen temperaturstabilisierten, geeichten Qualitätsquarz  als Taktgeber für die CMOS-Uhr eingebaut hat, müssen Sie mit einem Fehler von einigen Sekunden pro Tag rechnen. Durch Beobachtung über mehrere Tage können Sie diesen Fehler auch ohne aufwendige Messgeräte sehr gut bestimmen. Das Systemprogramm `clock` ermöglicht es Ihnen, den so bestimmten Fehler zu korrigieren.

Um die CMOS-Uhr zu "justieren", werden in der Datei `/etc/adjtime` die tägliche Abweichung und das Datum der letzten Korrektur eingetragen. Jedes Mal, wenn das Programm `clock` zum Justieren der CMOS-Uhr aufgerufen wird, errechnet es aus dem Datum der letzten Korrektur und dem Fehlerfaktor die aktuelle Abweichung und korrigiert sowohl die Systemzeit als auch die CMOS-Uhr, sobald der Fehler mehr als eine Sekunde beträgt. Außerdem wird in der Datei `/etc/adjtime` die Zeitmarke der letzten Korrektur aktualisiert und der nicht korrigierte Rest zur vollen Sekunde als dritter Wert eingetragen.

Wenn Sie zum Beispiel festgestellt haben, daß die CMOS-Uhr Ihres Rechners innerhalb von zwei Tagen um 23 Sekunden vorgeht, können Sie das System zur Korrektur folgendermaßen installieren:

```
[01] # echo "0.0 0 0.0" > /etc/adjtime
[02] # clock -au
[03] # date -s "Thu Aug 29 15:59:00 MET DST 1996"
Thu Aug 29 15:59:00 MET DST 1996
[04] # clock -wu
[05] # cat /etc/adjtime
0.000000 841345408 0.000000
[06] # echo "-11.500000 841345408 0.000000" > /etc/adjtime
[07] # _
```

Mit dem ersten Kommando wird die Datei `/etc/adjtime` neu erzeugt, durch das zweite Kommando wird der aktuelle Zeitstempel in die soeben erzeugte Datei eingetragen. Im dritten Kommando wird die Systemzeit "von Hand" mit der höchsten Genauigkeit gestellt. Durch das vierte Kommando wird die CMOS-Uhr mit der gerade eingestellten Systemzeit synchronisiert.

Für die Beispielrechnung beträgt der Fehler 23 Sekunden in zwei Tagen, das sind 11,5 Sekunden pro Tag. Das sechste Kommando zeigt, wie dieser Korrekturfaktor in die Datei `adjtime` eingetragen werden kann. Es ist wichtig, die beim zweiten Kommando erzeugte Zeitmarke (im Beispiel



841345408) dabei nicht zu verändern.

Wenn Sie jetzt das zweite Kommando oben in einem Initialisierungsscript beim Booten aufrufen, werden CMOS-Uhr und Systemzeit automatisch bei jedem Systemstart korrigiert.

Wenn die Systemuhr mit einer externen Zeitreferenz synchronisiert ist, wird die CMOS-Uhr automatisch alle 11 Minuten nachgestellt. Wenn Sie den während der Betriebspause aufgelaufenen Fehler beim Booten durch die oben beschriebene Methode korrigieren möchten, müssen Sie beim Shutdown dafür sorgen, daß die Zeitmarke in `/etc/adjtime` in etwa mit dem Datum des letzten Update der CMOS-Uhr vor dem Herunterfahren des Systems übereinstimmt.

## Linux und Echtzeit

Die beeindruckende Genauigkeit, mit der die Systemuhr von Linux synchronisiert werden kann, hat wenig mit Echtzeitfähigkeit zu tun. Bei einem Echtzeitsystem muß die Antwortzeit auf ein Ereignis in engen Grenzen vorhersagbar sein.

Der Scheduler von Linux unterstützt diese Funktionalität nicht. Das Scheduling wird spätestens durch einen Clocktick vom Timer ausgelöst. Bei einer Frequenz von 100Hz sind Verzögerungen bis zu 10 Millisekunden nicht zu vermeiden. Durch Interrupts, Prozesse mit höherer Priorität und Swapping kann es noch wesentlich länger dauern, bis ein Prozeß tatsächlich ausgeführt wird.

Weil Linux im Sourcecode vorliegt, läßt sich natürlich für spezielle Anwendungen eine Lösung programmieren. Beispielsweise kann der Scheduler modifiziert werden, oder eine zeitkritische Funktion wird als Interruptroutine in den Kernel integriert.

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [Linux als Mehrbenutzersystem](#) **Up:** [Systemverwaltung](#) **Previous:** [Prozeßordnung](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)



**Next:** [Genauere Betrachtung der Datensicherheit](#) **Up:** [Systemverwaltung](#) **Previous:** [Betrachtungen über die Zeit](#)

## Subsections

- [User, Gruppen und Allgemeinheit](#)
    - [Natürliche Personen und funktionale Rollen](#)
    - [Haupt- und Nebengruppen](#)
  - [Eigentum und Zugriffsrechte](#)
    - [Die Zugriffsrechte als Modus einer Datei](#)
    - [Bedeutung des Zugriffsmodus bei Verzeichnissen](#)
    - [Eigentum an Dateien](#)
    - [Eigentum an Prozessen](#)
    - [Wem gehört was - SUID und SGID Modus bei einem Programm](#)
    - [Besondere Modalitäten für Verzeichnisse](#)
    - [Mandatory Locking](#)
  - [Die Dateiattribute des ext2fs](#)
    - [Schreibzugriff nur zum Anhängen weiterer Daten](#)
    - [Einfrieren einer Datei](#)
    - [Geheime Daten gehören in den Reißwolf](#)
    - [Gespeichert ist noch nicht gesichert](#)
- 

# Linux als Mehrbenutzersystem

Für EDV-Anlagen im professionellen Einsatz ist es heutzutage selbstverständlich, daß mehrere Systembenutzer auf die gleichen Ressourcen zugreifen. Die Daten werden auf einem Fileserver oder in einer zentralen Datenbank gespeichert, Druckaufträge aus einem ganzen Netzwerk werden von einem einzigen Printserver bearbeitet.

Die Vorteile solch einer gemeinsamen Benutzung eines ganzen Rechnernetzes liegen auf der Hand:

1.  
Teure Hardware, die von einem einzigen Arbeitsplatz nicht ausgelastet wird, kann durch verteilte Nutzung effektiver eingesetzt werden.
2.  
Die Koordination von mehrgliedrigen Arbeitsprozessen wird durch die Benutzung eines gemeinsamen, zentralen Datenbestandes beschleunigt.
3.  
Information und Kommunikation in Arbeitsgruppen und ganzen Unternehmen werden durch ein gemeinschaftlich genutztes Netz verbessert.

# User, Gruppen und Allgemeinheit

Linux unterstützt das Arbeiten mit verteilten Systemen, indem es die natürlichen Konzepte von Individuum, Gruppe und Allgemeinheit fest in den Betriebssystemkern verankert hat.

Eine Person, die sich an einen Linux-Rechner setzt, um damit zu arbeiten, muß sich zuerst durch eine Login-Prozedur beim System anmelden und wird so als ``User" identifiziert. Alle Aktionen, die während einer ``Session" (der Zeitraum vom Login bis zum Logout) an diesem Rechner (Terminal) stattfinden, werden dem User zugeordnet. Alle User haben eigene Bereiche (Accounts) in denen sie allein über ihre eigenen Daten verfügen können.

Mehrere User, die bestimmte Systemressourcen oder Daten gemeinsam nutzen, können eine Gruppe bilden. Linux indentifiziert die Gruppenmitglieder entweder durch feste, namentliche Zuordnung oder durch eine vorübergehende Anmeldung, ähnlich der Login-Prozedur. Gruppen haben keine eigenen Accounts, können sich aber Daten aus den Bereichen beliebiger User teilen.

Linux organisiert seine ``Welt" hierarchisch, seine Betrachtungsweise ist eher differenzierend als vereinfachend. Deshalb wird der die Gruppe und das Individuum einschließende Oberbegriff der Allgemeinheit nicht direkt unterstützt, vielmehr wird den beiden ersten Begriffen derjenige der übrigen Systembenutzer gegenübergestellt. In der gleichen Weise wird übrigens auch der User von der Gruppe differenziert betrachtet.

User und Gruppen werden betriebssystemintern durch Nummern (User-ID und Group-ID) identifiziert. Die Zuordnung der IDs zu den verbalen Namen findet durch entsprechende Einträge in den Dateien `/etc/passwd` und `/etc/group` statt.

Sie können sich Ihre eigenen Identifikationsnummern mit dem Systemprogramm `id` anzeigen lassen.

## Natürliche Personen und funktionale Rollen

Die Begriffe User (Benutzer) und Group (Gruppe) sind dem wahren Leben entliehen und assoziieren das Bild von Menschen aus Fleisch und Blut. Da wir die Maschine Computer benutzen wollen um mit anderen Menschen zusammenzuarbeiten und zu kommunizieren, ist diese Bilder auch richtig.

Es bietet sich aber an -- und ist gängige Praxis bei allen echten Mehrbenutzer-Betriebssystemen, daß das Konzept von unterscheidbaren Usern und Gruppen auch für die Organisation der Systemdaten benutzt wird. Hier werden funktionale Rollen definiert, denen eigene Accounts und Gruppen zugeordnet werden.

Nach der Installation von Linux finden Sie in den Dateien `/etc/passwd` und `/etc/group` eine ganze Reihe solcher ``Pseudouser". Die wichtigste Rolle hat hier ein User namens `root` (Wurzel) mit der gleichnamigen Gruppe. User-ID und Group-ID von `root` sind 0 (Null). Die Rolle von `root` wird auch als *Superuser* bezeichnet. Sie zeichnet sich dadurch aus, daß sie mit den Rechten des Betriebssystems arbeitet. Alle Sicherheitsmechanismen, die das Betriebssystem zum Schutz der eigenen Integrität und zum Schutz der Benutzerprozesse und -daten eingebaut hat, sind für `root` außer Kraft gesetzt.

Weitere Pseudouser können für bestimmte Programmgruppen (beispielsweise News und Mail) oder für bestimmte Komponenten oder Gerätegruppen (beispielsweise Drucker, Band- und Diskettenlaufwerke) existieren.

# Haupt- und Nebengruppen

Das Betriebssystem verlangt, daß jeder User einer Gruppe angehört. Diese "Hauptgruppe" wird in der Benutzerdatenbank `/etc/passwd` eingetragen. Bei einigen Linux-Distributionen wird hier als Standard die gleiche allgemeine Gruppe für alle User benutzt (`users`, `user`, `other` oder ähnliche), bei anderen Distributionen wird von den Administrationstools beim Einrichten neuer Accounts für jeden neuen User eine eigene Gruppe mit dem Usernamen angelegt.

Festes Mitglied weiterer Gruppen können Sie werden, indem Sie von der Systemverwalterin in der Gruppendatenbank in den entsprechenden Mitgliederlisten eingetragen werden. Bei Gruppen, die durch ein Paßwort geschützt sind, können Sie auch ohne einen festen Eintrag in der Mitgliederliste vorübergehend die Gruppenmitgliedschaft erlangen, indem Sie sich mit dem Systemprogramm `newgrp` unter Angabe des korrekten Paßwortes in der Gruppe "einloggen".

Sie können sich Ihre gerade aktive Hauptgruppe und alle eingetragenen Nebengruppen mit dem Systemprogramm `id` anzeigen lassen.

## Eigentum und Zugriffsrechte

Durch die eindeutige Unterscheidbarkeit von Usern und Gruppen ist es naheliegend und möglich, den natürlichen Eigentumsbegriff auf alle Daten im System anzuwenden. Das wesentliche Recht des Eigentümers besteht darin, die Zugriffsrechte oder allgemeiner den Modus einer Datei bestimmen zu können. Die Gruppe und alle anderen User haben nur Rechte an einer Datei, sofern der Eigentümer ihnen diese Rechte abgibt.

In einem verteilten System, in dem private oder gruppeninterne Daten auf einem allgemein zugänglichen Medium gespeichert werden, kann der Eigentümer einer Datei durch entsprechende Zugriffsbeschränkung eine Veränderung oder das Lesen seiner Daten verbieten. Der Modus einer Datei kann für den Eigentümer, die Benutzergruppe und die sonstigen User separat und unabgänglich voneinander festgelegt werden. Dadurch lassen sich bereits mit Hilfe der Zugriffsrechte die Kompetenzen und Zuständigkeiten eines Gruppenarbeitsprozesses auf die Dateien, mit denen die Gruppe arbeitet, abbilden.

## Die Zugriffsrechte als Modus einer Datei

Linux unterscheidet drei Zugriffsmodi für Dateien:

### **lesbar (readable)**

Wenn eine Datei für Sie lesbar ist, können Sie sich deren Inhalt ansehen. Lesen bedeutet hier nicht unbedingt Verstehen. Die symbolische Abkürzung für lesbar ist `r`.

### **schreibbar (writable)**

Eine (be-)schreibbare Datei können Sie verändern. Sie können Daten in der Datei austauschen, an die Datei anhängen und aus der Datei entfernen. Sie können eine Datei, für die Sie Schreibrecht haben, sogar vollständig leeren; das Schreibrecht für die Datei erlaubt Ihnen aber nicht, die Datei vollständig aus dem Verzeichnis zu löschen. Die symbolische Abkürzung für schreibbar ist `w`.

### **ausführbar (executable)**

Wenn eine Datei für Sie ausführbar ist, können Sie diese Datei als Programm aufrufen und vom Betriebssystem ausführen lassen. Bei "echten" Programmen, die in Maschinensprache vorliegen und die direkt vom Betriebssystem ausgeführt werden können, benötigen Sie für die Ausführung einer Datei kein Leserecht. Bei interpretierten Programmen wie zum Beispiel Shellscripts oder

Perl-Programme benötigen Sie Leserecht für die Programmdatei, um sie ausführen zu können. Die symbolische Abkürzung für ausführbar ist `x`.

Der Eigentümer legt den Zugriffsmodus für jede seiner Dateien getrennt für sich selbst, die Benutzergruppe und die übrigen Systembenutzer fest. Beim Erzeugen einer Datei wird der Modus zuerst nach einem bestimmten Muster festgelegt, das jeder User durch die Funktion [umask](#) für seine Laufzeitumgebung einstellen kann. Der Modus einer existierenden Datei kann mit dem Systemkommando [chmod](#) verändert werden.

Sie können den Zugriffsmodus einer Datei mit dem Kommando `ls -l` anzeigen lassen. Im ersten Feld der Anzeige finden Sie die symbolischen Abkürzungen der Modi `readable`, `writable` und `executable` für den Eigentümer, die Gruppe und die anderen User (in dieser Reihenfolge).

```
[she@atlantis she]$ ls -l /home/buch/
total 298
-rw-r----- 1 hohndel autoren    4739 Mar  9 1995 X11R6.tex
-rw-r----- 1 hohndel autoren   44866 Dec 21 1995 XFree86.tex
-rw-rw-r--  1 she      she      56340 Dec 21 1995 grundlagen.tex
drwxr-xr-x  2 she      autoren   2048 Jun 28 11:22 man1
drwxr-xr-x  2 she      autoren   1024 Dec 21 1995 man8
-rw-rw----  1 okir     autoren  75735 Oct 24 1995 news.tex
-rw-rw-r--  1 she      she      63310 Dec 24 1995 sysadmin.tex
-rwxr-x---  1 she      autoren    51 Jun 28 19:04 twopage.letter
-rw-rw----  1 okir     autoren  48236 Oct 24 1995 uucp.tex
[she@atlantis she]$
```

Die Datei `X11R6.tex` im Beispiel ist für den Besitzer `hohndel` lesbar und kann von ihm verändert werden, für die Gruppe `autoren` ist sie nur lesbar und für alle anderen Systembenutzer ist sie nicht zugänglich.

Die Datei `sysadmin.tex` ist für den User und die Gruppe `she` lesbar und schreibbar, für alle anderen Systembenutzer ist sie nur lesbar.

Die Datei `news.tex` ist für den Eigentümer `okir` und für die Benutzergruppe `autoren` lesbar und schreibbar, für alle anderen User ist sie nicht zugänglich.

Die Datei `twopage.letter` ist für den Eigentümer `she` lesbar, schreibbar und ausführbar, für die Gruppe der Autoren ist sie lesbar und ausführbar und für alle anderen User ist sie nicht zugänglich.

## Bedeutung des Zugriffsmodus bei Verzeichnissen

Verzeichnisse werden in vielerlei Hinsicht wie die anderen Dateitypen behandelt, die Bedeutung der Modi für die Zugriffsrechte können aus offensichtlichen Gründen nicht einfach übernommen werden. Die "Übersetzung" der Wirkung von Zugriffsrechten auf die Verzeichnisse sieht folgendermaßen aus:

### **lesbar**

Bei den Verzeichnissen bedeutet die Lesbarkeit, daß die berechtigten Personen den Inhalt des Verzeichnisses sehen können. Ohne dieses Recht können Programme wie `ls` kein Listing des Verzeichnisses anzeigen.

### **schreibbar**

Wenn ein Verzeichnis schreibbar ist, können Dateien darin angelegt, umbenannt und gelöscht werden. Das Löschen einer Datei ist auch erlaubt, wenn diese Datei selbst nicht verändert werden

darf.

## ausführbar

Die Ausführbarkeit eines Verzeichnisses erlaubt es den berechtigten Usern, in dieses Verzeichnis als aktuelles Verzeichnis zu wechseln und auf die Dateien darin zuzugreifen. Wenn ein Verzeichnis ausführbar, aber nicht lesbar ist, kann auf die Dateien oder Unterverzeichnisse ``blind" zugegriffen werden.

## Eigentum an Dateien

Linux unterscheidet sechs verschiedene Dateiarten, die im Dateisystem gespeichert werden: normale Files, Verzeichnisse, Gerätedateien, Sockets, FIFOs und Links. All diese Objekte werden bei ihrer Erzeugung in den Datenstrukturen des Dateisystems eingetragen. In diesen Einträgen (den I-Nodes) wird unter anderem die Information über den Eigentümer und die Benutzergruppe, der die Datei zugeordnet wird, festgehalten. Jede Datei kann nur einem Eigentümer und einer Benutzergruppe gehören.

Das Programm `ls` zeigt Eigentümer und Gruppe jeder Datei an, wenn es wie im Beispiel oben mit der Option `-l` aufgerufen wird. Die dritte Spalte zeigt den Namen des Eigentümers, die vierte Spalte zeigt den Namen der Gruppe.

## Eigentum an Prozessen

Linux betrachtet nicht nur die mehr oder weniger ``festen" Daten auf einem dauerhaften Speichermedium als Objekte, die einem Eigentümer zugeordnet werden. Der Eigentumsbegriff wird in gewisser Weise auch auf die im Speicher befindlichen Daten und die laufenden Prozesse angewendet.

Jedes Kommando, das ein User über die Tastatur eingibt, erzeugt einen Prozeß im Arbeitsspeicher des Rechners. Im normalen Betrieb befinden sich immer mehrere Prozesse gleichzeitig im Speicher, die vom Betriebssystem streng voneinander abgegrenzt werden. Die einzelnen Prozesse werden mit allen Daten, die in ihrem virtuellen Adressraum enthalten sind, einem User als Eigentümer zugeordnet.

Die Eigentümer der Prozesse werden vom Programm `ps` angezeigt, wenn es mit der Option `-u` aufgerufen wird.

USER	PID	%CPU	%MEM	SIZE	RSS	TTY	STAT	START	TIME	COMMAND
bin	89	0.0	1.0	788	328	?	S	13:27	0:00	rpc.portmap
hohndel	190	0.0	2.0	1100	628	3	S	13:27	0:00	-bash
hohndel	613	0.0	1.3	968	424	3	S	15:05	0:00	vi XF86.tex
nobody	167	0.0	1.4	932	444	?	S	13:27	0:00	httpd
okir	191	0.0	2.0	1100	628	4	S	13:27	0:00	-bash
okir	622	0.3	1.5	1064	476	4	S	15:05	0:00	elm
root	1	0.0	1.0	776	316	?	S	13:27	0:03	init [3]
root	2	0.0	0.0	0	0	?	SW	13:27	0:00	(kflushd)

# Wem gehört was - SUID und SGID Modus bei einem Programm

Die einfache Grundregel für die Bestimmung des Eigentümers eines Objektes lautet: jedes Ding gehört dem User, der es erzeugt hat. Die Gruppe eines Objektes richtet sich nach der aktuellen Hauptgruppe des Eigentümers zum Zeitpunkt der Entstehung der Datei.

Auf dieser Regel aufbauend gibt es ein paar Besonderheiten und Ausnahmen, die die Leistungsfähigkeit und Flexibilität des Betriebssystems deutlich verbessern.

Wie bereits erwähnt, betrachtet Linux auch die laufenden Prozesse als Objekte mit Eigentümern. Ein Prozeß entsteht, indem ein User ein Programm aus einer Programmdatei aufruft. Es ist naheliegend, den User, der ein Programm startet, als den Eigentümer des dadurch entstehenden Prozesses anzusehen.

Das ist aber nicht die einzig mögliche Sichtweise. Es kann das berechtigte Interesse des Eigentümers einer Programmdatei sein, daß der Prozeß, der durch die Ausführung dieser Datei entsteht, auch sein Eigentum ist. So eine Situation tritt häufig dann ein, wenn ein Programm Systemdienste (Services) für alle Systembenutzer anbietet.

Der Eigentümer einer Programmdatei kann deshalb den Ausführungsmodus einer Programmdatei so verändern, daß das Betriebssystem den daraus entstehenden Prozeß mit seiner effektiven User-ID oder mit der effektiven Group-ID seiner Benutzergruppe ausführt. Der Prozeß einer Programmdatei mit S-Modus hat damit die Rechte des Eigentümers (beziehungsweise der Gruppe) der Datei. Zusätzlich behält jeder Prozeß mit einer veränderten User- oder Gruppen-ID auf einer zweiten Ebene auch die realen Identitäten und Rechte der aufrufenden Person. Damit hat so ein Prozeß die Freiheit, in beiden Bereichen zu arbeiten. Ein *Set User ID Programm* bildet auf diese Weise ein *intelligentes Tor* zwischen den ansonsten hermetisch voneinander abgeriegelten Bereichen zweier User.

Das Angebot eines solchen Tores setzt das Vertrauen des Eigentümers in die Zuverlässigkeit des Programmes voraus, das mit seinen eigenen Rechten laufen soll. Der *Größte Anzunehmende Unfall* bei der Verwendung eines SUID-Programms würde eintreten, wenn der Benutzer eine interaktive Shell mit den Rechten des Dateieigentümers bekommen würde. Insbesondere bei Programmen, die mit der User-ID von `root` laufen, können aber auch *kleinere* Fehlfunktionen des Programms Schaden an Systemdaten oder an den Dateien anderer User anrichten.

Die Systemprogramme, die bei der Installation einer der bekannten und aktuellen Linux-Distributionen in den SUID-Modus gesetzt werden, sind gut getestet und nach dem gegenwärtigen Wissensstand zuverlässig und sicher.

Der SUID-Modus wird wie alle anderen Modi einer Datei mit dem Systemprogramm [chmod](#) verändert.

Sie erkennen Programme, die mit den Rechten des Eigentümers oder der Gruppe der Programmdatei arbeiten, in einem langen Listing von `ls` durch die symbolischen Abkürzungen `s` anstelle von `x` für normal ausführbare Dateien.

## Besondere Modalitäten für Verzeichnisse

Es gibt eine weitere Ausnahme von der Zuordnung des Eigentums an Dateien nach dem *Verursacherprinzip*: der Eigentümer eines Verzeichnisses kann bestimmen, daß die in diesem Verzeichnis erzeugten Dateien der gleichen Benutzergruppe gehören wie das Verzeichnis selbst. Das geschieht, indem das Verzeichnis den S-Modus für die Gruppe (*Set Group-ID*) bekommt.

Die Zugriffsrechte auf ein Verzeichnis werden durch das SGID-Bit nicht verändert. Um eine Datei in einem solchen Verzeichnis anzulegen, muß ein User das Schreibrecht in der für ihn zutreffenden

Kategorie (Eigentümer, Gruppe, andere User) haben. Wenn ein User zum Beispiel weder der Eigentümer noch Mitglied der Benutzergruppe eines SGID-Verzeichnisses ist, muß das Verzeichnis für die ``anderen User" beschreibbar sein. Die in dem SGID-Verzeichnis erzeugte Datei gehört dann der Gruppe des Verzeichnisses, auch wenn der User selbst dieser Gruppe nicht angehört.

Der SGID-Modus verändert nur das Verhalten des Betriebssystems beim Erzeugen neuer Dateien. Der Umgang mit bereits existierenden Dateien ist in diesen Verzeichnissen völlig normal. Das bedeutet beispielsweise, daß eine Datei, die außerhalb des SGID-Verzeichnisses erzeugt wurde, beim Verschieben dorthin ihre originale Gruppe behält (wohingegen sie beim Kopieren die Gruppe des Verzeichnisses bekommen würde).

Auch das Programm `chgrp` arbeitet in SGID-Verzeichnissen völlig normal: der Eigentümer einer Datei kann sie jeder Gruppe zueignen, der er selbst angehört. Gehört der Eigentümer nicht zu der Gruppe des Verzeichnisses, kann er die Datei mit `chgrp` nicht dieser Gruppe zueignen -- dazu muß er sie in dem Verzeichnis neu erzeugen.

Es ist zwar möglich, auch bei einem Verzeichnis den S-Modus für den Eigentümer (*Set User-ID*) zu setzen, diese Einstellung hat aber keine Wirkung. Das Betriebssystem erlaubt es den Usern nicht, Dateien an andere User ``zu verschenken".

Linux unterstützt noch einen weiteren Spezialmodus für Verzeichnisse, bei dem das Löschen oder Umbenennen von darin enthaltenen Dateien nur mit ausreichenden Rechten an der jeweiligen Datei selbst vorgenommen werden können.

Mit diesem ``*T-Modus*" kann einem Problem begegnet werden, das bei der gemeinsamen Verwendung öffentlicher Verzeichnisse entstehen kann: das Schreibrecht für das Verzeichnis erlaubt auch das Löschen fremder Dateien, unabhängig vom Zugriffsmodus für die Datei.


Beispielsweise sind die `tmp/-`Verzeichnisse ``öffentlicher Raum", in dem von vielen Programmen temporäre Dateien angelegt werden. Um darin Dateien anlegen zu können, haben alle User für diese Verzeichnisse Schreibrecht. Damit hat jeder User auch das Recht, Dateien in diesem Verzeichnis zu löschen. Normalerweise betrachtet das Betriebssystem beim Löschen oder Umbenennen einer Datei die Zugriffsrechte auf die Datei selbst nicht weiter.

Indem die `tmp/-`Verzeichnisse in den T-Modus gesetzt werden, können nur noch der Eigentümer und die zum Schreiben in der Datei berechtigten User eine dort gespeicherte Temporärdatei löschen.

## Mandatory Locking

Im Mehrbenutzerbetrieb kommt es gelegentlich vor, daß mehrere Prozesse auf die gleiche Datei zugreifen. Das Betriebssystem unterstützt diese Betriebsart und erlaubt sogar, daß mehrere Prozesse gleichzeitig die selbe Datei zum Schreiben geöffnet halten. Wenn beide Prozesse Veränderungen an der gleichen Stelle in der Datei vornehmen, führt das zu unvorhersagbaren Ergebnissen.

Um dieses Problem zu lösen, haben sich mehrere Methoden zum Sperren von Dateien entwickelt. Die traditionelle Form der Sperrung durch Lockfiles und auch die Sperrung durch die Systemcalls `flock` und `fcntl` sind *kooperative Sperren*. Das bedeutet, daß jeder Prozeß, der eine Datei verändern will, von sich aus prüfen muß, ob eine Sperre eines anderen Prozesses vorhanden ist um gegebenenfalls auf die Veränderung der Datei vorübergehend zu verzichten.

Linux kann eine weitere Form der Dateisperrung unterstützen: das *Mandatory Locking*  (Verbindliche Sperrung). Wenn eine Datei oder ein Teil davon mit so einer Sperre belegt ist, wird jeder Versuch, diesen Teil zu verändern, vom Betriebssystem abgebrochen.

Der Mechanismus zum verbindlichen Sperren benutzt den Systemcall `fcntl` im Stil von System V. Damit eine Datei mit diesem Systemaufruf verbindlich gesperrt wird, muß bei einer für die Gruppe nicht ausführbaren Datei das SGID-Bit gesetzt sein.

Das folgende Beispiel zeigt, wie Sie eine Datei zum verbindlichen Sperren kennzeichnen:

```
[she@atlantis she]$ chmod g+s-x database
[she@atlantis she]$ ls -l database
-rw-rwSr--  1 she      she          92160 Jul  9 09:51 database
[she@atlantis she]$
```

Das große S im Feld für die Ausführungsrechte der Gruppe zeigt, daß die Datei nicht ausführbar und das SGID-Bit gesetzt ist.

## Die Dateiattribute des ext2fs

Zusätzlich zu der aus dem Dateisystemkonzept von Unix übernommenen Methode, die Datensicherheit durch Reglementierung der Zugriffsrechte zu erhöhen, bietet das ext2fs mit seinen Dateiattributen weitere Mechanismen an, die den Datenschutz in verschiedenen Richtungen verbessern.

1.

Dateien lassen sich seit ext2-0.5a durch die Attribute `a` (append) und `i` (immutable) zusätzlich vor Veränderungen schützen.

2.

Dateien können durch das Attribut `s` (secure) beim Löschen durch zufällige Daten überschrieben und dadurch zuverlässig vernichtet werden.

3.

Die besonders sensiblen Metadaten einer Datei können im Speicher und auf der Festplatte synchron verwaltet werden, indem das Attribut `S` gesetzt wird.

Alle Dateiattribute können mit dem `chattr`-Kommando verändert werden. Das Kommando `lsattr` zeigt die aktuellen Attribute ähnlich wie das `ls`-Programm die Zugriffsrechte.

## Schreibzugriff nur zum Anhängen weiterer Daten

Beim traditionellen System der Zugriffsregelung wird das Schreiben ganz allgemein entweder erlaubt oder verboten. Die Erlaubnis erstreckt sich uneingeschränkt auf das Schreiben an jeder beliebigen Stelle der Datei. Damit bedeutet Schreiben in diesem Sinne sowohl Schreiben als auch Überschreiben.

Das Betriebssystem kann, dem POSIX-Standard entsprechend, zwischen wahlfreiem Schreiben und dem ausschließlichen Schreiben am Ende einer Datei, also dem Anhängen von Daten, unterscheiden.

Das ext2-Dateisystem unterstreicht diese Unterscheidung und führt damit ein zusätzliches Sicherheitsmerkmal ein. Durch das append-Attribut `a` wird jeder Schreibzugriff automatisch am Ende der Datei ausgeführt; es gibt dann keine Möglichkeit, die Schreibposition an eine andere Stelle zu setzen.

Zusätzlich kann eine durch dieses Attribut gesicherte Datei nicht gelöscht, gelinkt, umbenannt oder verschoben werden. Dieser Schutz läßt sich auch durch Rootprivilegien nicht umgehen.

Das append-Attribut kann jeder Benutzer für seine eigenen Dateien ändern. Mit Rootprivilegien ist auch



das Ändern fremder Dateiattribute möglich.

## **Einfrieren einer Datei**

Allein die Superuserin kann eine Datei im ext2-Dateisystem völlig einfrieren. Das immutable-Attribut verbietet jeden Schreibzugriff auf eine Datei. Zusätzlich sind wie beim append-Attribut Löschen, Linken, Umbenennen und Verschieben der Datei unmöglich.

Der normale Eigentümer einer Datei kann das immutable-Attribut nicht verändern. Damit kann ihm durch dieses Attribut jede Veränderung wirksam verboten werden. Diese Restriktion macht für natürliche Systembenutzer keinen Sinn. Für news, mail und ähnliche Systemaccounts, die sich dadurch auszeichnen, daß viele Programme zur Laufzeit mit den Rechten dieser ``Eigentümer" arbeiten, bietet immutable wirksam zusätzliche Sicherheit vor Mißbrauch.

Das immutable-Attribut selbst kann mit Rootprivilegien wieder gelöscht werden. Einen absoluten Schutz vor unberechtigter Veränderung einer Datei bietet also auch dieses Attribut nicht.

## **Geheime Daten gehören in den Reißwolf**

Beim Löschen einer Datei wird in den Linux-Dateisystemen normalerweise der Verzeichniseintrag gelöscht und die Inode sowie die Datenblöcke freigegeben, wenn der Verzeichniseintrag der letzte für diese Datei war.

Die freigegebenen Datenblöcke werden bei irgendeiner Gelegenheit durch einen anderen Prozeß wieder belegt. Es ist kein Geheimnis, daß der Inhalt solcher Datenblöcke von dem Prozeß, dem sie zugeteilt worden sind, auch gelesen werden kann. Wenn ein Prozeß den Datenblock liest, bevor er eigene Daten hineingeschrieben hat, kann er fremde Daten darin finden.

Um diese unbeabsichtigte Weitergabe von Daten zu verhindern, bietet das ext2-Dateisystem die Möglichkeit, alle von einer Datei belegten Datenblöcke durch zufällige Zeichen zu überschreiben, bevor sie beim Löschen für andere Prozesse freigegeben werden. Das secure-Attribut schickt die Dateien beim Löschen also durch einen elektronischen Reißwolf.

Sie sollten allerdings bedenken, daß häufig bei der Bearbeitung von Dateien durch andere Programme Kopien angelegt werden, die nicht automatisch mit dem secure-Attribut ausgestattet sind.

## **Gespeichert ist noch nicht gesichert**

Indem Linux einen Großteil des freien Arbeitsspeichers als Puffer für die relativ langsamen Festplattenzugriffe nutzt, wird die Geschwindigkeit des System spürbar erhöht.

Diese erhebliche Leistungssteigerung geht zu Lasten der Datensicherheit. Wenn das System nämlich aus irgendeinem Grund abstürzen sollte, sind die im Puffer veränderten, aber noch nicht auf die Festplatte zurückgeschriebenen Daten verloren.

Besonders unangenehm wird es, wenn es sich bei den verlorenen Daten um Strukturinformation handelt. In diesem Fall kann auch die bereits sicher auf der Festplatte befindliche Information unzugänglich werden.

Der Verzicht auf die Datenpufferung kommt nicht ernsthaft in Betracht. Ein Betriebssystem, das mehrere

Benutzer und viele Prozesse gleichzeitig bedienen soll, braucht ein sehr schnelles Dateisystem.

Um wenigstens die Sicherheit der bereits auf Festplatte gespeicherten Daten zu optimieren, bietet das ext2-Dateisystem die Möglichkeit, mit dem synchron-Attribut die ungepufferte Verwaltung der Metadaten einer Datei zu erreichen. Verzeichnis und Inode werden unmittelbar nach ihrer Veränderung auf die Festplatte geschrieben.

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [Genauere Betrachtung der Datensicherheit](#) **Up:** [Systemverwaltung](#) **Previous:** [Betrachtungen über die Zeit](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

## Subsections

- [Authentifizierung](#)
  - [Zusätzliche Sicherheit durch Datenverschlüsselung](#)
    - [Pretty Good Privacy \(PGP\)](#)
    - [DES-Verschlüsselung und Crypto-Filesystem](#)
- 

# Genauere Betrachtung der Datensicherheit


Neben der Betriebssicherheit, die vor allem durch Qualitätsmängel der billig produzierten Hardware, durch Bedienfehler und durch fehlerhafte Software beeinträchtigt wird, steht besonders die Sicherheit der wertvollen und vertraulichen Daten vor Verlust und unberechtigttem Zugriff im Mittelpunkt der Aufmerksamkeit einer erfolgreichen Systemverwaltung.

Die beste Methode, Daten vor dem Verlust zu schützen, ist die Sicherung einer oder mehrerer Kopien auf Magnetbändern oder anderen geeigneten Medien. Dem Thema Backups ist ein eigenes Kapitel in diesem Buch gewidmet.

Im folgenden Abschnitt soll das Problem der Sicherheit von Daten vor unberechtigttem Zugriff genauer betrachtet werden.

## Authentifizierung

Um die Eigentumsrechte und das damit verbundene System der verschiedenen Zugriffsrechte für Eigentümer, Besitzergruppen und übrige Systembenutzer durchzusetzen, muß jede Person, die sich an den Rechner setzt und damit arbeiten will, vom System als User identifiziert und zugelassen werden. Das geschieht durch die Login-Prozedur: bevor das Betriebssystem bereit ist, ein Programm für einen Benutzer zu starten, muß dieser seinen Benutzernamen eingeben und seine Identität durch die verdeckte Eingabe eines geheimen Passwortes bestätigen.

Die ersten acht Zeichen des Passwortes werden durch einen nicht umkehrbaren Algorithmus  codiert und mit einem gespeicherten Schlüsselwort verglichen. Dieses Schlüsselwort ist zu einem früheren Zeitpunkt durch die gleiche Funktion aus dem Passwort erzeugt und in eine Datei geschrieben worden. Stimmen das gespeicherte und das soeben erzeugte Schlüsselwort überein, müssen auch die Passwörter übereingestimmt haben. Das System geht davon aus, daß das geheime Passwort nur dem berechtigten Benutzer bekannt ist und akzeptiert deshalb die Person an der Tastatur als User.

Die codierten Passwörter werden zusammen mit den anderen Benutzerdaten in der Datei

/etc/passwd gespeichert. Der Datensatz wird bei der Einrichtung eines neuen Accounts von der Systemverwalterin erzeugt. Die Paßwortdatei ist für alle Systembenutzer lesbar, kann aber nur mit Rootprivilegien verändert werden. Weil aus dem Schlüsselwort das ursprüngliche Paßwort nicht auf einfachem Wege erzeugt werden kann, ist diese Methode der Authentifizierung grundsätzlich sicher. Gelegentlich auftauchende Erfolgsgeschichten von Hackern, die sich unberechtigten Zugang zu Rechnern verschafft haben, zeigen aber auch die Grenzen dieses einfachen Sicherheitssystems auf.

Wenn Sie meinen, Ihr Linux-System müsse höheren Sicherheitsstandards genügen, erhalten Sie auf den nächsten Seiten ein paar Anregungen für dessen Verbesserung. Sie sollten sich aber keine Illusionen machen: die Sicherheit einer Computeranlage hängt von mehr als der Software und deren Konfiguration ab.

Bevor Sie sich daran machen, Ihren Rechner ``dicht'' zu machen, sollten Sie sich über die Ziele klar werden und die Verhältnismäßigkeit der Mittel nicht außer Acht lassen. Um bestimmte Daten vor unberechtigtem Zugriff zu schützen ist es nicht unbedingt nötig, das gesamte System abzuschließen.

Beachten Sie auch, daß jeder ``Angreifer'' bei dem Versuch an Ihre Daten heranzukommen den Weg des geringsten Widerstandes gehen wird. Sie können sich eine Menge Bemühungen um die Optimierung der Authentifizierung sparen, wenn Sie nicht verhindern, daß jemand, dem Sie nicht trauen, Zugang zur Hardware bekommt und beispielsweise einen Neustart mit einer eigenen Bootdiskette durchführt.

Vor diesem Hintergrund wird die Verbesserung des Authentifizierungssystems erst Sinnvoll, wenn Sie Ihren Linux-Rechner in irgendeiner Weise vernetzen.

Es gibt einige Schwachstellen des einfachen Authentifizierungssystems, die es einer Person ermöglichen können, sich die Rechte eines anderen Users anzueignen.

Jeder User hat die Qual der Wahl bei der Festlegung eines neuen Paßwortes. Wer kennt nicht die Angst, das soeben kunstvoll ausgedachte Paßwort bis zum nächsten Einloggen zu vergessen? Diese Angst verursacht die beiden wichtigsten Schwachstellen des einfachen Authentifizierungssystems: triviale Paßwörter und schriftliche Notizen in der Nähe des Rechners.


Namen, Autokennzeichen, Telefonnummern und Geburtsdaten sind nicht geheim und können deshalb schnell erraten werden.

Der Hang zu einfachen Paßwörtern ermöglicht es den Hackern, ein uncodiertes Paßwort aus dem Schlüsselwort zu gewinnen, wenn es sich um ein echtsprachliches Wort oder eine leichte Variation davon handelt. Dazu wird ein schneller Computer eingesetzt, um systematisch ganze Wörterbücher abzuarbeiten und sämtliche Wörter vorwärts und rückwärts, in Groß- und Kleinschreibung und mit einer ganzen Reihe weiterer Variationen auszuprobieren.

Theoretisch läßt sich durch ``rohe Gewalt'', also den Einsatz massiver Rechenleistung, jedes Paßwort durch Ausprobieren knacken. Die Methode mit den Wörterbüchern ist aber um die Größenordnung vieler Zehnerpotenzen schneller.

Zunächst ist das Problem der schwachen Paßwörter natürlich das desjenigen Users, der solch ein schwaches Paßwort verwendet. Das Problem läßt sich durch eine Veränderung im System deutlich entschärfen, indem die Schlüsselwörter nicht mehr für alle Systembenutzer lesbar in der Benutzerdatenbank /etc/passwd gespeichert werden. Die Benutzerdatenbank selbst muß lesbar bleiben, weil die meisten der darin enthaltenen Daten (Benutzername, User-ID, Realname, Heimatverzeichnis,...) im alltäglichen Arbeitsbetrieb häufig von den Systembenutzern abgefragt werden. Da bei diesen Zugriffen auf die Benutzerdatenbank das codierte Paßwort nie benötigt wird,

ist es möglich, die codierten Paßwörter in einer separaten Datei abzulegen, die nur durch ein paar besonders privilegierte Programme gelesen werden kann.


Das Shadow-Paßwort-System, das Sie als Ergänzung zu den meisten Linux-Distributionen bekommen können,  arbeitet genau nach dieser Methode.

Die codierten Paßwörter werden hier in der Datei `/etc/shadow` gespeichert, die nur mit Superuserrechten gelesen werden kann. Die Programme, die die Login-Prozedur durchführen, haben diese Rechte und können deshalb ohne Probleme damit arbeiten.


Das Shadow-Paßwort-Paket hat noch einige Features, die die Sicherheit des Authentifizierungssystems weiter verbessern sollen. So kann das Programm, das es den Usern erlaubt, ihr Paßwort zu ändern, bestimmte Schwächen feststellen und solche schwachen Paßwörter ablehnen. Durch einen "Alterungsmechanismus" können die User gezwungen werden, ihr Paßwort nach einer gewissen Zeit zu verändern, sie können aber auch gezwungen werden, ihr Paßwort eine gewisse Zeit zu behalten.

Ob eine solche Maßregelung sinnvoll oder inakzeptabel ist, müssen Sie selbst entscheiden. Eine absolute Sicherheit vor Paßwortknackern bietet auch die Verdoppelung des Schutzes nicht.

Ein Sicherheitsproblem, das von der Shadow-Paßwort-Suite nicht gelöst wird, taucht in allen vernetzten Systemen auf, in denen sich User über das Netz auf anderen Rechnern einloggen. Hierbei muß bei dem bisher beschriebenen Authentifizierungssystem das unverschlüsselte Paßwort zum Zielrechner übertragen werden. Auch wenn es nicht als Echo auf dem Bildschirm erscheint, wird es doch Zeichen für Zeichen auf dem Ethernetkabel oder auf der Telefonleitung transportiert. Solche Verbindungen können abgehört werden, und jemand kann sich auf diese Weise in den Besitz von unverschlüsselten Paßwörtern bringen.

Mit dem Programmpaket `deslogin`  können Sie diese Sicherheitslücke schließen. Das Programm `deslogin` kann ähnlich wie `telnet` oder `rlogin` benutzt werden. Wie die beiden letztgenannten Programme baut der `deslogin`-Client eine Netzwerkverbindung zu einem Serverdämon auf einem entfernten Rechner auf. Client und Server führen eine Authentifizierung durch, bei der das Paßwort ähnlich wie beim Kerberos-Protokoll in einem "Challenge-Response" Dialog nur in verschlüsselter Form ausgetauscht wird. Zusammen mit dem Paßwort wird auch ein zufällig erzeugter Schlüssel übertragen, mit dem der weitere Datenverkehr zwischen Client und Server verschlüsselt wird. Damit schützt `deslogin` nicht nur das geheime Paßwort, sondern sämtliche über das Netzwerk übertragenen Daten. Wie der Name schon andeutet, benutzt `deslogin` als Verschlüsselungsfunktion die im *Data Encryption Standard* definierten Methoden.

`deslogin` arbeitet mit einer eigenen Paßwortdatei, die ebenfalls mit DES verschlüsselt ist und vom Server nur gelesen werden kann, wenn dieser beim Start den korrekten Schlüssel erhalten hat. Die Paßwörter selbst sind nicht auf 8 signifikante Zeichen beschränkt, können also beliebig lang sein.

Wenn aus irgendwelchen Gründen die Verwendung von `deslogin` nicht möglich ist, können Sie wenigstens Ihr geheimes Paßwort vor dem Ausspionieren schützen, indem Sie ein System mit Einwegpaßworten verwenden. Das S/Key-Paket  läßt sich leicht in das Shadow-Paßwort-System integrieren und bietet diese Funktionalität. S/Key liefert bei der Frage nach dem Paßwort zwei Parameter (eine Zahl und ein Wort). Diese beiden Parameter müssen vom Benutzer zusammen mit seinem geheimen Paßwort und mit Hilfe eines sicheren Computers zu einem Schlüssel verarbeitet werden, der dann über das Netz an das Login-Programm geschickt wird und den Zugang zu dem entfernten System öffnet.

Weil S/Key beim nächsten Loginversuch einen anderen Parameter liefert, und natürlich auch ein anderes Schlüsselwort erwartet, ist das ``Belauschen" der Login-Prozedur völlig wertlos.

## **Zusätzliche Sicherheit durch Datenverschlüsselung**

Es gibt Fälle, in denen alle bisher beschriebenen Methoden der Datensicherung nicht ausreichend sind. Ein Sicherheitssystem ist immer nur so stark, wie sein schwächster Punkt. Wenn jemand durch Einbruch oder Diebstahl Zugang zur Hardware bekommt, ist jede Authentifizierung überflüssig: der Eindringling kann sich bequem Superuser-Rechte und damit den Zugriff zu allen Daten auf den Festplatten verschaffen.

Diese und ähnliche Überlegungen zeigen deutlich die Grenzen der Sicherung vertraulicher Daten vor unberechtigtem Zugriff durch Mechanismen des Betriebssystems. Diese Mechanismen sind primär dazu entworfen, die Integrität des Systems sicherzustellen und die Systembenutzer vor versehentlichen oder fahrlässigen Beschädigungen fremder und auch eigener Daten zu schützen.

Wenn Sie mit Papieren umgehen, die Sie in einem Stahlschrank oder gar in einem Safe aufbewahren, dann sollten Sie die elektronischen Versionen dieser Dokumente nicht den einfachen Schutzfunktionen des Betriebssystems überlassen. Wenn Sie die Festplatte mit den vertraulichen Daten nicht jeden Tag ausbauen und über Nacht in den Safe schließen möchten, können Sie Ihre elektronischen Dokumente verschlüsseln und damit einen Safe auf Ihrer Festplatte einrichten.

Sie haben unter Linux mehrere Verschlüsselungsfunktionen zur Verfügung, die den Vergleich mit dem Safe im Keller einer Bank nicht zu scheuen brauchen. Die Verschlüsselungssysteme sind in Deutschland kostenlos und ohne Probleme per FTP zu bekommen, sie sind leicht zu installieren und einfach zu bedienen. Wenn Sie diese Systeme sorgfältig benutzen, brauchen Sie sich um die Sicherheit Ihrer Daten keine Sorgen mehr zu machen.

### **Pretty Good Privacy (PGP)**

Das von Phil Zimmerman entwickelte Verschlüsselungsprogramm PGP gehört zu den sichersten Cryptographischen Systemen auf dem Markt. Es ist schwierig, die Qualität eines Verschlüsselungsprogramms zu beurteilen. Unter den zivilen Cryptoanalytikern genießen die beiden Algorithmen, die PGP benutzt, jedenfalls hohes Ansehen und es mag als Indiz für die Qualität des Systems gelten, daß die Sicherheitsbehörden und Geheimdienste der USA und anderer Länder die Verbreitung des Programms zu verhindern versuchten.

Die staatlichen Dienste mußten feststellen, daß ihre Aktivitäten zur Eindämmung wirksamer Verschlüsselungstechnologie mehr zu deren Publicity und damit zur weiteren Verbreitung beigetragen haben. Deshalb wurde das Strafverfahren gegen Phil Zimmerman eingestellt.

Dieser scheinbaren Liberalisierung stehen auf der anderen Seite neue Gesetze gegenüber, die den Geheimdiensten jederzeit den unbemerkten Zugang zu den unverschlüsselten Daten sichern soll. In diesem Sinne enthält der in vielen Hardwarelösungen verwendete Clipper-Chip eine Hintertür, die es den Geheimdiensten erlaubt, an die unverschlüsselten Daten heranzukommen. PGP steht nicht im Verdacht, so eine Hintertür zu haben.

Die Besonderheit, die PGP speziell für den Einsatz mit vernetzten Computern interessant macht, liegt im asymmetrischen Verschlüsselungsverfahren (RSA). Dieses Verfahren eignet sich für den Austausch von Nachrichten, beispielsweise per E-Mail, weil es das bei symmetrischen Verschlüsselungsverfahren auftretende Problem der Schlüsselübergabe löst. Bei PGP kann ein öffentlicher Schlüssel (Public Key) gefahrlos allgemein zugänglich gemacht werden, und ein zweiter, geheimer Schlüssel erlaubt nur dessen Besitzer die Entschlüsselung einer mit dem Public Key verschlüsselten Nachricht.

Umgekehrt kann der Absender eines Briefes seine Nachricht mit Hilfe seines geheimen Schlüssels signieren. Der Empfänger kann dann mit dem Public Key die Authentizität des Absenders und die Integrität der Nachricht prüfen.

Das Programm `pgp` selbst hat kein besonders ansprechendes User-Interface. Speziell für den Einsatz mit E-Mail gibt es aber eine "aufgebohrte" Version des Mailprogramms `elm`, die Ver- und Entschlüsselung sowie die Pflege des "Schlüsselrings" weitgehend automatisiert.

Für die Verschlüsselung von "Massendaten" benutzt PGP einen Algorithmus namens IDEA. Dieses Verfahren gilt als sehr stark. Es ist stärker und schneller als das RSA-Verfahren, arbeitet aber mit symmetrischen Schlüsseln. Deshalb wird bei der Verschlüsselung von Nachrichten der eigentliche Text mit IDEA und einem zufällig erzeugten Einwegschlüssel verschlüsselt, der Schlüssel wird dann mit RSA und dem Public- bzw. dem Secret-Key verschlüsselt und das ganze dann zu einem Paket verschnürt. Beim Empfänger wird dann zuerst der Schlüssel mit dem RSA-Verfahren entschlüsselt und damit dann der Text der Nachricht.

Wenn Sie Dateien auf dem lokalen System mit PGP verschlüsseln wollen, können Sie IDEA direkt mit einem von Ihnen selbst gewählten Schlüssel verwenden.

Sie finden die Sourcen zu PGP auf verschiedenen FTP-Servern im Internet, beispielsweise im Verzeichnis `/pub/os/linux/sources/kernel/net-source/base` auf `ftp.informatik.hu-berlin.de` unter dem Namen `pgp-2.6.2i-5.tar.gz`.

## DES-Verschlüsselung und Crypto-Filesystem

Die Abkürzung DES steht für *Data Encryption Standard*, es handelt sich um ein in den 70er Jahren von IBM unter Mitwirkung der US National Security Agency entwickeltes Verschlüsselungsverfahren.

DES arbeitet mit einem 56-Bit kleinen Schlüssel, was 1977 als einigermaßen sicher gelten konnte. Heute muß davon ausgegangen werden, daß es Behörden und größeren Firmen mit vertretbarem Aufwand möglich ist, einen DES-Schlüssel durch rechnergestützte Suche herauszufinden. Trotzdem wird DES noch häufig zur Sicherung vertraulicher Daten eingesetzt.

Die weite Verbreitung von DES und dessen Unterstützung durch viele Anwendungen machen DES auch in Gegenwart besserer Verschlüsselungstechnologien interessant. Es nützt der beste Safe leider nichts, wenn seine Bedienung so unbequem ist, daß er von denjenigen, die ihn benutzen sollen, nicht akzeptiert wird.

Sie finden DES Implementationen per FTP beispielsweise auf `ftp.gwdg.de` im Verzeichnis `/misc/crypt/cryptography/symmetric/des/` oder auf `concert.cert.dfn.de` im Verzeichnis `/pub/tools/crypt/des`. Beide Sites bieten aktuelle und gut gepflegte Archive zum Thema Verschlüsselung.

Außer dem bereits beschriebenen Programm [deslogin](#) ist besonders das Cryptographische

Filesystem (CFS) für Linux interessant. Sie finden die offizielle Version von Olaf Kirch auf [ftp.mathematik.th-darmstadt.de](ftp.mathematik.th-darmstadt.de/pub/linux/okir/cfs-1.1.2.tar.gz) im Verzeichnis `/pub/linux/okir/cfs-1.1.2.tar.gz`

CFS arbeitet wie das Network File System (NFS). Durch den CFS-Server wird ein Verzeichnis im Arbeitsbereich eines Users exportiert und über die Netzwerkfunktionen des Betriebssystems mit dem normalen NFS-Protokoll an einer anderen Stelle in das Dateisystem eingebunden. Der CFS-Server filtert alle Datenströme zwischen dem per NFS eingebundenen Verzeichnis und dem realen Verzeichnis im Arbeitsbereich. Dabei werden die Daten in der einen Richtung verschlüsselt, in der anderen Richtung werden sie wieder entschlüsselt.

Die Daten und sogar die Datei- und Verzeichnisnamen in dem ``realen'' Verzeichnis auf der Festplatte sind nur durch den Filter zu sehen. Das mit dem NFS-Protokoll eingebundene virtuelle Verzeichnis ist nur für den Eigentümer lesbar. Selbst mit Rootrechten ist ein Zugriff auf die Daten im virtuellen Verzeichnis nicht möglich, weil das NFS die Privilegien des Superusers nicht respektiert.

Eine andere Variante eines Verschlüsselten Dateisystems wird vom Loop-Device angeboten. Wenn die Algorithmen zur DES-Verschlüsselung in den Kernel einkompiliert werden, kann das Loop-Device die Daten in der von ihm verwalteten Datei mit dieser Funktion verschlüsseln. Die notwendigen Kernelpatches und die Hilfsprogramme finden Sie auf verschiedenen FTP-Servern.

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [Benutzer eintragen](#) **Up:** [Systemverwaltung](#) **Previous:** [Linux als Mehrbenutzersystem](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)



## Subsections

- [Eintrag in /etc/passwd](#)
  - [Gruppenzwang](#)
  - [Das Heimatverzeichnis anlegen](#)
- 

# Benutzer eintragen

Die Notwendigkeit einer Benutzerverwaltung liegt bei Mehrbenutzersystemen auf der Hand. Die *User* müssen dem System durch einen Eintrag in der Benutzerdatenbank `/etc/passwd` bekannt gemacht werden. Nachdem die Installation einer neuen Linux-Distribution abgeschlossen ist, gibt es zunächst nur den `root`-Account für die Systemverwalterin. Alle weiteren Benutzerbereiche müssen erst eingerichtet werden.

Selbst wenn Linux nur von einer einzigen Person benutzt wird, ist unbedingt anzuraten, diesen User als 'normalen' Benutzer einzutragen. Beim alltäglichen Arbeiten unter Rootrechten sind die meisten Sicherheitsvorkehrungen des Kernels ganz abgeschaltet, oder sie lassen sich leicht umgehen. Auf diese Weise können leicht Fehler entstehen, die das gesamte System betreffen. Sie vermeiden die Gefahr, indem Sie sich unter einem normalen Useraccount einloggen und nur vorübergehend mit dem Programm `su` eine Shell mit Rootrechten starten.

Bei vielen Linux-Distributionen gibt es zur Unterstützung der Systemverwalterin Dienstprogramme, die das Ein- und Austragen von Benutzern mehr oder weniger automatisieren. Bei Caldera und RedHat ist es das Programm `usercfg`, bei Unifix und Linux-FT ist es `xadmin`, bei SuSE `yast` und bei wieder anderen sind es die Programme `useradd`, `userdel`, `groupadd` und `groupdel`, die mit der Shadow-Paßwort-Suite installiert werden. Die ersten der genannten Programme arbeiten mit grafischer Oberfläche und sind weitgehend selbsterklärend. Ansonsten können Sie Details zur Benutzung in den Manualpages nachlesen.

Der Vorgang bei der Einrichtung eines neuen User-Accounts ist bei allen Programmen im Prinzip gleich:

1.  
es wird ein Eintrag in der Paßwortdatei `/etc/passwd` angelegt
2.  
wenn nötig wird ein Eintrag in die Gruppendatei `/etc/group` erzeugt
3.  
das Heimatverzeichnis wird erzeugt, eine ``Grundausstattung'' hineinkopiert und alles dem neuen User übereignet
4.  
wenn nötig wird der User noch in weiteren Listen eingetragen, zum Beispiel für Disk-Quotas,

Alle Dateien, die beim Einrichten eines neuen Accounts bearbeitet werden, sind normale Textfiles. Sie können jeden Schritt ohne weiteres "von Hand" beziehungsweise mit Hilfe eines Texteditors durchführen.

## Eintrag in `/etc/passwd`

Die zentrale Benutzerdatenbank ist die Datei `/etc/passwd`. In dieser Datei muß jeder Anwender aufgeführt sein, damit er sich einloggen kann.

Für jeden Anwender muß die Systemverwalterin hier eine Zeile mit sieben Feldern anlegen. Die Felder werden durch einen Doppelpunkt `:` voneinander getrennt. Sie haben folgende Bedeutung:

*Benutzername:Paßwort:Benutzernummer:Gruppennummer:GCOS:Heimat:Shell*

- Das erste Feld enthält den Benutzernamen. Dieser Name darf aus beliebigen druckbaren Zeichen bestehen. Allerdings ist es sinnvoll und üblich, nur Kleinbuchstaben für die Benutzernamen zu verwenden.
- Das zweite Feld enthält das verschlüsselte Paßwort. Wenn dieses Feld leer bleibt, ist der Account durch kein Paßwort gesichert, also frei zugänglich. Wenn anstelle eines korrekt verschlüsselten Paßwortes ein Klartextwort in die Datenbank geschrieben wird, ist unter dem Benutzernamen kein Login möglich. Wenn Sie das Shadow-Paßwort-System verwenden, ist das Paßwort-Feld hier typischerweise durch den Buchstaben `x` gesperrt. Das verschlüsselte Paßwort ist stattdessen in der Datei `/etc/shadow` gespeichert, die nur vom Superuser und von privilegierten Prozessen gelesen werden kann.

Jedem Benutzer steht das `passwd`-Kommando zur Verfügung, um sein Paßwort selbst zu verändern.

Beim manuellen Neueintrag eines Benutzers wird dieses Feld zunächst freigelassen.

Unmittelbar nachdem der Account eingerichtet ist sollte der Benutzer oder die Systemverwalterin unter dem neuen Benutzernamen das `passwd` Kommando aufrufen und ein Paßwort eingeben, damit der Account vor unberechtigten Usern geschützt ist.


- Das dritte Feld enthält die Benutzernummer (UID) . Die Nummer ist eine beliebige nicht negative Zahl (bis 64000). Es ist üblich, den "natürlichen Systembenutzern" User-IDs zu geben, die sich deutlich von denen der "funktionalen Rollen" `bin`, `daemon` usw. unterscheiden. Beispielsweise können alle zusätzlichen User Benutzernummern größer als 100 erhalten. Die halbautomatischen Programme zur Benutzerverwaltung suchen sich selbst die nächste freie Nummer aus dem Bereich, der von den Programmierern vorgesehen wurde.

Die Benutzer werden kernelintern nicht durch die Namen, sondern durch die ID unterschieden. Wenn eine Nummer zweimal mit unterschiedlichen Benutzernamen vergeben wird, behandelt der Kernel intern die beiden Accounts völlig identisch. Bei den Kommandos, die einen Benutzernamen anzeigen (z.B. `ls -l` oder `id`), wird immer der Eintrag verwendet, der als erstes in `/etc/passwd` steht.

- Das vierte Feld enthält die Nummer (GID) einer Benutzergruppe. Die hier angegebene Gruppe ist die Standardgruppe des neuen Users. Bei Caldera und RedHat wird von `usercfg` für jeden neuen User automatisch eine eigene Gruppe angelegt, die die gleiche ID hat wie der

Useraccount selbst. Bei anderen Distributionen wird eine bestimmte Gruppe, beispielsweise `users`, als gemeinsame Standardgruppe für alle User eingetragen.

Jeder User muß durch die Zuordnung in der Datei `/etc/passwd` Mitglied mindestens einer Benutzergruppe sein. Zusätzlich kann jeder User beliebig vielen weiteren Gruppen angehören. Die weiteren Zuordnungen finden durch entsprechende Einträge in der Datei `/etc/group` statt.

- Der fünfte Eintrag ist das sogenannte GCOS-Feld. Es enthält in der Regel den Realnamen des Benutzers, es können aber noch zusätzliche Daten zur Person oder zur Systemverwaltung in diesem Feld enthalten sein. 

Der Realname wird von vielen News&Mail-Programmen bei der Zusammenstellung der Absenderadresse benutzt.

- Das sechste Feld enthält den absoluten Pfadnamen des Heimatverzeichnisses des Anwenders. Dieses Verzeichnis ist der persönliche Bereich des Benutzers, in dem er seine eigenen Dateien aufbewahren kann. Üblicherweise erhält ein neuer User von der Systemverwalterin einige Profil- und RC-Dateien als Erstausrüstung in das Heimatverzeichnis.
- Das siebente Feld enthält den Namen des Programms, das von `login` nach erfolgreicher Anmeldung gestartet werden soll. Der Benutzer kann mit dem Programm `chsh` diesen Eintrag selbst ändern. Die möglichen Programme (Shells) sind in der Datei `/etc/shells` aufgelistet.

Wenn ein Benutzer keine interaktive Shell haben soll, kann auch ein beliebiges anderes Programm mit allen Argumenten in dieses Feld eingetragen werden. Zum Beispiel kann hier für UUCP Accounts der `uucico` Daemon aufgerufen werden. Der Account „`sync`“ ist ein anderes Beispiel, mit dem Sinn, auch von außerhalb die Synchronisation des Dateisystems mit dem Blockdepot auslösen zu können.

Das letzte Feld kann auch leer bleiben. Dann wird automatisch die Standardshell `/bin/sh` gestartet. Das siebente Feld wird nicht durch einen Doppelpunkt, sondern durch ein Zeilenende abgeschlossen.

Bereits mit der Erstinstallation eines Linux-Systems sind in der Paßwortdatei eine Reihe von Einträgen enthalten. Die mit diesen Daten festgelegten Benutzerbereiche gehören nicht zu natürlichen Personen, sondern zu funktionalen Rollen des Systems. Die Pseudouser sind als Eigentümer von Dateien und Verzeichnissen hilfreich, um die mit dem Eigentum an Systemdaten verbundenen Zugriffsrechte flexibel an die speziellen Anforderungen anzupassen. Auf diese Weise sorgen die Accounts `uucp` für das UUCP-Datentransfersystem oder `postgres` für das gleichnamige Datenbanksystem dafür, daß auf die Daten, mit denen sie operieren, nur unter der Kontrolle entsprechender Serverprogramme zugegriffen werden kann.

Darüberhinaus können die Pseudouser noch dazu dienen, die unüberschaubare Masse der Systemdaten ein wenig zu ordnen. Diese Möglichkeit wird jedoch von den Linux-Distributoren nicht genutzt: mehr als 99% aller Systemdateien gehören ohne Notwendigkeit dem Superuser `root`.


## Gruppenzwang

Das Betriebssystem unterstützt Gruppenarbeit indem es dem Eigentümer einer Datei erlaubt, die

Zugriffrechte einer Benutzergruppe auf diese Datei frei festzulegen. Durch den oben beschriebenen Eintrag in der Datei `/etc/passwd` wird jeder User einer Benutzergruppe fest zugeordnet. Auf diese Weise wird festgelegt, zu welcher Gruppe die Dateien des Users bei deren Erzeugung gehören.

Jeder User kann zusätzlich noch weiteren Gruppen angehören. Diesen Gruppen kann er seine eigenen Dateien nach belieben zuordnen. Dabei ist jedoch zu beachten, daß jede Datei nur einer einzigen Gruppe gehören kann.

Ähnlich wie die Benutzeraccounts müssen auch die Gruppen dem Betriebssystem bekannt gemacht werden, bevor sie benutzt werden können. Das geschieht durch einen Eintrag in der Datei `/etc/group`. Für jede Benutzergruppe existiert eine Zeile in dieser Datei. Jede Zeile besteht aus vier Feldern, die durch einen Doppelpunkt (':') voneinander getrennt sind.

- Das erste Feld enthält den Namen der Gruppe.
- Das zweite Feld kann das verschlüsselte Paßwort für den Gruppenzugang enthalten.
- Das dritte Feld enthält die Gruppennummer (GID). Die Nummer ist eine beliebige nichtnegative Zahl bis 64000 (255) . Jede Gruppennummer darf nur einmal verwendet werden.
- Das vierte Feld schließlich enthält eine durch Kommata getrennte Liste der Namen aller Gruppenmitglieder.

Für einen gültigen Eintrag muß mindestens das erste und dritte Feld (Gruppenname und Gruppennummer) vorhanden sein. Durch so einen Eintrag wird der Gruppennummer, die in der Paßwortdatei einem User zugeordnet wurde, ein Gruppenname gegeben. Dieser Name wird bei einem langen Listing vom Programm `ls` ausgegeben.

Die Felder für Paßwort und/oder Userliste müssen nur für die Gruppen ausgefüllt werden, deren Mitglieder mehr als einer Benutzergruppe angehören. Wenn ein User in einer oder mehr Userlisten aufgeführt ist, hat er automatisch die Zugriffsrechte dieser Gruppen, zusätzlich zu der Hauptgruppe, die in der Paßwortdatei eingetragen ist. Die Gruppenzugehörigkeit einer neu erzeugten Datei wird weiterhin durch die Hauptgruppe bestimmt. Der User kann seine eigenen Dateien mit Hilfe des Systemprogramms `chgrp` jeder Gruppe übereignen, in deren Userliste er eingetragen ist. Außerdem ermöglicht das Programm `newgrp` das vorübergehende Wechseln der Hauptgruppe.

Die in der Userliste eingetragenen Gruppenmitglieder werden nicht nach einem Paßwort gefragt, wenn sie mit dem `newgrp`-Kommando die Hauptgruppe wechseln wollen. Anwender ohne Eintrag können sich mit einem Paßwort legitimieren, um die Rechte einer bestimmten Gruppe zu erhalten, wenn im zweiten Feld des Gruppeneintrags ein verschlüsseltes Paßwort eingetragen ist und das Programm `newgrp` diese Art der Authentifizierung unterstützt.

Weil Gruppenpaßwörter zwangsläufig ausgetauscht werden müssen, wird aus Gründen der Datensicherheit von der Verwendung dieser Methode abgeraten. Anstelle eines verschlüsselten Paßwortes sollte zur Absicherung ein Sperreintrag (z. B. `VOID` oder `*`) in das zweite Feld geschrieben werden.

Wie in der Paßwortdatei sind auch in der Gruppendatei bereits mehrere Gruppen eingerichtet, wenn Sie ein Linux-System neu installieren. Der Zweck dieser Gruppen entspricht dem der Pseudouser in `/etc/passwd`. Insbesondere für die Zugriffskontrolle auf die verschiedenen Gerätedateien in `/dev` werden einige Gruppen tatsächlich benutzt. Die Gruppenbezeichnungen `disk`, `tty`, `daemon`, `mail` u.s.w. erklären sich selbst. Die Gruppe `wheel`, die unter BSD alle User enthält, die das Systemprogramm `su` ausführen dürfen, wird von Linux in dieser Weise nicht unterstützt.

Wie bei der Paßwortdatei gibt es auch für die Gruppendatei eine Erweiterung durch das Shadow-Paßwortsystem. Die Gruppenpaßwörter, die in der normalen group Datei wie in passwd verschlüsselt, aber für alle User lesbar abgelegt sind, werden hier dort der separaten Datei /etc/gshadow gespeichert. Dort werden auch zusätzliche Informationen zur Gruppe festgehalten, beispielsweise der Name des zum Ein- und Austragen von Mitgliedern autorisierten Gruppenverwalters.

## Das Heimatverzeichnis anlegen

Jedem natürlichen Benutzer muß durch den Eintrag in der Paßwortdatei ein Verzeichnis als eigener Bereich zugeordnet werden. Dieses *Heimatverzeichnis* trägt häufig den Namen des Benutzers und wird normalerweise als Unterverzeichnis von /home angelegt. Das Verzeichnis muß dem Benutzer und seiner Hauptgruppe gehören.

Wenn der Useraccount nicht durch eines der halbautomatischen Administrationsutilities erzeugt wird, muß das Verzeichnis von der Systemverwalterin mit dem Systemprogramm mkdir erzeugt und anschließend durch chown dem neuen Eigentümer ``geschenkt" werden.

Häufig werden einem neuen User einige Dateien als Erstausrüstung in sein Heimatverzeichnis kopiert. Eine Sammlung solcher Dateien befindet sich im Verzeichnis /etc/skel. Es handelt dabei Initialisierungsdateien für verschiedene Programme, zum Beispiel .bash\_profile, .Xdefaults, .fvwmrc oder .emacs. Da die Namen dieser Dateien mit einem Punkt beginnen, werden sie bei einem Listing nur angezeigt, wenn die Option -a für *alle* benutzt wird. Die Dateien in /etc/skel sollten vor allem als Beispiel und Anregung für den User dienen.

Das folgende Beispiel zeigt die Kommandofolge zum Erzeugen eines neuen Heimatverzeichnisses:

```
# mkdir /home/dirk
# cp -a /etc/skel/{.[^.]*,*} /home/dirk
# chown -R dirk.users /home/dirk
# chmod 1700 /home/dirk
# _
```

Die Konstruktion { .[^.]\*,\* } wird von der bash ausgewertet und sorgt dafür, daß wirklich der gesamte Inhalt des Verzeichnisses kopiert wird. Die Dateien, deren Namen mit einem Punkt beginnen, werden von einem einfachen Sternchen nicht erfaßt. Die beiden Links . und . . zeigen auf das Verzeichnis selbst und sein übergeordnetes Verzeichnis und sollen deshalb nicht kopiert werden.

Durch die Änderung des Zugriffsmodus auf den Wert 1700 wird das neue Heimverzeichnis für die Gruppe und alle anderen Systembenutzer vollständig abgeschlossen. Das Löschen einer Datei aus diesem Verzeichnis ist nur dem Eigentümer möglich.

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [Partitionen und Dateisysteme](#) **Up:** [Systemverwaltung](#) **Previous:** [Genauere Betrachtung der Datensicherheit](#)



## Subsections

- [Die Festplatte partitionieren](#)
    - [Hintergrundinformation](#)
    - [Die Partitionstabelle](#)
    - [Benutzung von fdisk](#)
    - [Die Bedeutung des Boot-Flags](#)
    - [Die Bedeutung des Partitionstyps](#)
  - [Das Dateisystem einrichten](#)
  - [Das Dateisystem zusammenbauen](#)
    - [Die Dateisystemtypen von Linux 2.0](#)
    - [Allgemeine Optionen beim Mounten der Dateisysteme](#)
    - [Gemeinsame Optionen für verschiedene Dateisystemtypen](#)
    - [Spezielle Optionen für das EXT2FS](#)
    - [Spezielle Optionen für die FAT-Dateisysteme](#)
    - [Spezielle Optionen für das iso9660 Dateisystem](#)
    - [Spezielle Optionen für Netzwerk-Dateisysteme \(NFS und SMB\)](#)
    - [Feste Vorgaben für die Zusammensetzung des Dateisystems](#)
  - [Die Konsistenz des Dateisystems prüfen](#)
    - [Das Rootfilessystem reparieren](#)
- 

# Partitionen und Dateisysteme

Die Installation eines Linux-Systems geschieht in den seltensten Fällen zum Selbstzweck. Nach der Erstinstallation beginnt eine mehr oder weniger intensive Benutzung des Systems. Damit ist eine stetige Veränderung des Dateisystems verbunden: die Benutzer legen eigene Daten darin ab, die Anforderungen an das System erfordern die Installation zusätzlicher Programme, Mail und News drohen, den Rest des freien Festplattenplatzes aufzufüllen. In jedem Fall füllt sich jede noch so große Festplattenpartition nach und nach mit Daten. Früher oder später erfordern diese "Abnutzungserscheinungen" administrative Eingriffe in die bestehende Struktur des Dateisystems.

Wenn der Plattenplatz knapp wird, gibt es für die Systemverwalterin mehrere Möglichkeiten zu reagieren. Zuerst können überflüssige Daten gelöscht und selten benötigte Daten komprimiert werden. Wenn die Heimatverzeichnisse der Benutzer die gesamten Ressourcen verschlingen, kann das Problem durch Einrichtung von Disk-Quotas eingeschränkt werden. Schließlich kann das Dateisystem vergrößert werden, um den erweiterten Anforderungen gerecht zu werden.

Es gibt verschiedene Möglichkeiten, ein Dateisystem zu vergrößern:



- In Zeiten ständig sinkender Hardwarepreise kann eine zusätzliche Festplatte installiert werden, um einen Teil der Daten des bestehenden Systems aufzunehmen.
- In einem lokalen Netzwerk können die von mehreren Systemen gemeinsam benutzten Daten auf einem Server zentralisiert und über das Network File System (NFS) netzweit verteilt werden.
- Selten benötigte, statische Daten können auf eine CD oder eine Wechselplatte geschrieben und nur bei Bedarf in das Dateisystem eingebunden werden.
- Eventuell können auch Daten auf die Partition eines anderen Betriebssystems umgelagert werden.

## Die Festplatte partitionieren

Beim PC muß eine Festplatte immer in Partitionen eingeteilt werden. Wenn Sie für die Erstinstallation oder zur Erweiterung Ihres Systems eine leere Festplatte benutzen, ist die Partitionierung unproblematisch. Wenn Sie jedoch die Partitionen einer bereits teilweise belegten Festplatte ändern wollen, müssen Sie unbedingt alle Daten der existierenden Installation auf einem anderen Medium sichern. Bei jeder Veränderung der bestehenden Partitionierung gehen mit sehr hoher Wahrscheinlichkeit die bisher auf der Festplatte gespeicherten Daten verloren.

## Hintergrundinformation

Eine Festplatte besteht aus mehreren speziell beschichteten Aluminiumscheiben. Die Beschichtung kann magnetisiert werden und dadurch Daten speichern, ganz ähnlich wie ein Tonband Musik speichert. Zum Schreiben und Lesen gibt es die Schreib/Leseköpfe (oder einfach **Köpfe**), für jede beschichtete Scheibenseite einen. Diese Köpfe sind alle zusammen an einem Arm befestigt, der alle gemeinsam (in radialer Richtung) auf den Scheiben bewegen kann. Die Bewegung erfolgt in sehr genau bestimmten Schritten. Bei jedem Schritt erreicht jeder Kopf eine **Spur** der Scheibe. Die Gesamtheit aller Spuren, die von den parallelen Köpfen überstrichen wird, heißt **Zylinder**.

Um einen möglichst schnellen und direkten Zugriff auf die Daten zu ermöglichen, werden die Spuren nicht kontinuierlich beschrieben. Der Festplattencontroller richtet beim Formatieren **Sektoren** ein, in denen jeweils 512 Bytes Platz haben. Diese Sektoren können einzeln ``adressiert" werden, das bedeutet, daß jeder Sektor einzeln gelesen und beschrieben werden kann. Die Betriebssysteme verwalten aber in der Regel immer zwei Sektoren gemeinsam als einen **Block** von 1024 Bytes.

Die konkrete ``Plattengeometrie" ist herstellerabhängig. Bei den meisten Festplatten werden die Daten im Setupmenü eingegeben und im CMOS gespeichert. Bei den SCSI-Festplatten übernimmt der Hostadapter die unteren Ebenen der Gerätesteuerung, deshalb müssen diese Platten nicht im CMOS des Rechners eingetragen werden.

Aus verschiedenen Gründen ist es sinnvoll, die gesamte Kapazität einer Festplatte in verschiedene **Partitionen** zu unterteilen:

1.

können manche Betriebssysteme nur relativ kleine Festplatten in einem Stück verwalten,

2.

können sich verschiedene Betriebssysteme eine Festplatte teilen, indem jedes eine eigene



Partition erhält,

3.

läßt sich der Datenbestand auf der Festplatte besser strukturieren,

4.

erhöht sich die Datensicherheit, weil von einem Plattenfehler oft nur eine Partition betroffen wird.

## Die Partitionstabelle

Die Einteilung der Festplatte in Partitionen ist nicht an irgendwelche physikalischen Gegebenheiten gebunden, sondern wird allein durch die Vereinbarung in der **Partitionstabelle** festgelegt. Wenn Sie sich an ein paar Regeln halten, wird die Partitionstabelle von allen PC-Betriebssystemen akzeptiert.

Der erste Sektor der Festplatte ist der **primäre Bootsektor**. In diesem Sektor kann eine Partitionstabelle für höchstens vier Partitionen angelegt werden. Diese Partitionen heißen **primäre Partitionen**. Anstelle einer primären Partition kann in dem primären Bootsektor auch (genau) eine **erweiterte Partition** definiert werden, die den gesamten Plattenplatz enthält, der keiner primären Partition zugeordnet ist. In der erweiterten Partition können weitere **logische Partitionen** eingerichtet werden, die im Prinzip genauso aufgebaut sind wie die primären Partitionen, mit dem Unterschied, daß nur von den primären Partitionen direkt gebootet werden kann. Mit einem Hilfsprogramm wie dem Linux Loader (LILO) kann das Betriebssystem von beliebigen Partitionen, auch auf anderen Festplatten, gebootet werden, allerdings muß LILO selbst immer auf einer primären Partition der ersten Festplatte installiert werden (möglicherweise auch auf der erweiterten Partition). Weitere Hinweise dazu finden sich in der Dokumentation, die mit dem LILO-Paket verteilt wird.

Um bei der geteilten Nutzung der Festplatte durch mehrere Betriebssysteme Fehlinterpretationen der Partitionstabelle zu vermeiden, sollten Sie die Partitions Grenzen immer auf Zylindergrenzen legen. Wenn Sie die Standardeinstellung von `unit` bei `fdisk` nicht verändern, werden die Partitions Grenzen automatisch auf Zylindergrenzen gelegt.

## Benutzung von fdisk

Das `fdisk`-Programm von Linux dient zur Erstellung von Linux-Partitionen auf Festplatten. Um DOS Partitionen einzurichten, sollte das gleichnamige Programm für DOS verwendet werden.

Wenn `fdisk` ohne Parameter aufgerufen wird, bearbeitet es die Partitionstabelle der ersten Festplatte, das ist `/dev/hda`. Um die zweite Festplatte, `/dev/hdb`, zu partitionieren, muß das Programm als `'fdisk /dev/hdb'` aufgerufen werden. Die erste SCSI Platte wird entsprechend als `'/dev/sda'` angesprochen.

Wenn es korrekt aufgerufen wurde, meldet sich `fdisk` mit dem Prompt:

```
Command (m for help): _
```

`fdisk` erwartet einen einzelnen Buchstaben gefolgt von einem RETURN als Kommando. Das Kommando `'m'` zeigt folgendes Menü:

```
Command action
a   toggle a bootable flag
```

```

c  toggle the dos compatiblity flag
d  delete a partition
l  list known partition types
m  print this menu
n  add a new partition
p  print the partition table
q  quit without saving changes
t  change a partition's system id
u  change display/entry units
v  verify the partition table
w  write table to disk and exit
x  extra functionality

```

Das **print**-Kommando zeigt die aktuelle Partitionstabelle an. Eine typische Anzeige sieht beispielsweise so aus:

```

Disk /dev/hda: 15 heads, 17 sectors, 1001 cylinders
Units = cylinders of 255 * 512 bytes

```

Device	Boot	Begin	Start	End	Blocks	Id	System
/dev/hda1		1	1	402	51246+	6	DOS 16-bit >=32M
/dev/hda2	*	403	403	550	18870	81	Linux/MINIX
/dev/hda3		551	551	920	47175	83	Linux native
/dev/hda4		921	921	1001	10327+	82	Linux swap

In diesem Beispiel werden 4 Partitionen angezeigt. Es sind vier Primärpartitionen angelegt. Von den Partitionen hda2 kann gebootet werden.

Das **unit**-Kommando ändert die Einheiten, in denen die Partitionstabelle angezeigt und bearbeitet wird. Das Kommando schaltet zwischen Sektoren und Zylindern als Einheit um. Weil MS-DOS die Partitionen auf kompletten Zylindern erwartet, ist es sinnvoll, mit der (voreingestellten) Zylindereinheit zu arbeiten.

Die Anzeige in Sektoren soll trotzdem am gleichen Beispiel gezeigt werden:

```

Disk /dev/hda: 15 heads, 17 sectors, 1001 cylinders
Units = sectors of 1 * 512 bytes

```

Device	Boot	Begin	Start	End	Blocks	Id	System
/dev/hda1		1	17	102509	51246+	6	DOS 16-bit >=32M
/dev/hda2	*	102510	102510	140249	18870	81	Linux/MINIX
/dev/hda3		140250	140250	234599	47175	83	Linux native
/dev/hda4		234600	234600	255254	10327+	82	Linux swap

Der Beginn einer Partition muß nicht mit dem Start des Datenbereichs übereinstimmen, wie das Beispiel der MS-DOS Partition zeigt, bei der die ersten Sektoren für Verwaltungsdaten freigehalten werden. Das Linux-*fdisk* setzt den Start des Datenbereichs normalerweise auf den Partitionsbeginn.

Die auf die Partitionsgröße folgenden '+' Zeichen markieren Partitionen mit ungerader Sektorzahl. Weil Linux immer Blocks zu zwei Sektoren, also immer mindestens 1024 Bytes belegt, bleibt auf diesen Partitionen ein Sektor ungenutzt.

Das **verify**-Kommando überprüft die Partitionstabelle auf Fehler und gibt die Anzahl der nicht belegten Sektoren aus.

Das **quit**-Kommando beendet *fdisk*. Die Veränderungen an der Partitionstabelle werden nicht automatisch geschrieben. Eine Veränderung an der Partitionstabelle wird erst durch ein *write*-Kommando auf der Festplatte gesichert. Es ist möglich, *fdisk* zu jedem Zeitpunkt mit **CTRL-C** zu

unterbrechen.

Das **delete**-Kommando löscht eine Partition aus der Partitionstabelle. Die von dieser Partition belegten Sektoren werden freigegeben. Alle auf diesen Sektoren gespeicherten Daten gehen dabei normalerweise verloren. Das *delete*-Kommando fragt nach der Partitionsnummer, die es löschen soll. Soll doch lieber keine Partition gelöscht werden, kann *fdisk* mit `CONTROL-C` beendet werden.

Wenn eine nicht existierende Partition angegeben wird, erscheint eine Fehlermeldung. Wenn die erweiterte Partition gelöscht wird, verschwinden automatisch alle logischen Partitionen auf der erweiterten Partition mit.

Wenn eine logische Partition gelöscht wird, werden alle darüberliegenden logischen Partitionen sofort neu nummeriert, so daß alle logischen Partitionen immer fortlaufende Nummern haben.

Das **new** (add) Kommando erlaubt das Anlegen einer neuen Partition. Voraussetzung zum Anlegen einer neuen Partition ist natürlich, daß noch freie Sektoren existieren.

- Wenn ein Platz in der primären Partitionstabelle frei ist und Sektoren außerhalb einer eventuell existierenden erweiterten Partition frei sind, kann eine Primärpartition angelegt werden.
- Wenn ein Platz in der primären Partitionstabelle frei ist und noch keine erweiterte Partition existiert, kann eine erweiterte Partition angelegt werden.
- Wenn innerhalb der erweiterten Partition Sektoren frei sind, kann eine logische Partition angelegt werden.

Wenn mehr als eine Möglichkeit zum Anlegen einer neuen Partition existiert, wird gefragt, ob eine primäre, erweiterte oder logische Partition angelegt werden soll. Wenn eine primäre oder die erweiterte Partition angelegt werden soll, muß eine freie Partitionsnummer im Bereich 1-4 gewählt werden.

Nun muß der Beginn der neuen Partition angegeben werden. Dazu erscheint beispielsweise folgende Meldung:

```
First cylinder (403-1001): _
```

Die Zahlen bezeichnen den ersten und den letzten freien Zylinder. Es müssen nicht alle Zylinder in dem angegebenen Bereich frei sein, weil innerhalb des Bereichs noch eine komplette andere Partition liegen kann. Die Angabe eines bereits belegten Zylinders wird einfach abgelehnt, und es wird erneut nach dem ersten Zylinder der neuen Partition gefragt.

Wenn ein gültiger Zylinder für den Beginn der Partition bestimmt ist, wird nach dem letzten Zylinder gefragt:

```
Last cylinder (403-1001): _
```

Hierbei sind alle Zylinder in dem angegebenen Bereich erlaubt.

Wenn nicht alle freien Zylinder belegt worden sind, kann das Kommando erneut aufgerufen werden.

Bevor die veränderte Partitionstabelle gesichert wird, sollte auf jeden Fall das *verify*-Kommando aufgerufen werden.

Die veränderte Partitionstabelle wird in jedem Fall nur durch ein ausdrückliches *write*-Kommando auf die Festplatte geschrieben. In diesem Fall erscheint die Aufforderung, den Rechner neu zu starten:

The partition table has been altered.  
Please reboot before doing anything else.

Es ist ohne Problem möglich, *fdisk* sofort wieder zu starten, um weitere Veränderungen an der Partitionierung vorzunehmen.

**ACHTUNG!** Es darf auf keinen Fall eines der Programme *mkfs*, *mkswap*, *mount* oder *swapon* aufgerufen werden, bevor der Rechner neu gestartet wurde!

## Die Bedeutung des Boot-Flags

Die meisten modernen Rechner haben ihr Betriebssystem nicht mehr in permanenten Speicherbausteinen auf der Hauptplatine, sondern laden erst nach dem Einschalten ein Betriebssystem in den Arbeitsspeicher. Und selbst dieser Ladevorgang findet nicht auf eine ganz bestimmte Art und Weise statt, sondern wird von einem Bootprogramm ausgeführt. Einzig der Aufruf dieses Bootprogramms ist durch das BIOS festgelegt:

Wenn es nicht ausdrücklich im BIOS-Setup anders bestimmt ist, wird im ersten Diskettenlaufwerk nach einer Diskette gesucht. Auf dieser Diskette wird aus dem ersten Sektor (in dem sich auch die Partitionstabelle befindet) das Bootprogramm geladen und ausgeführt.

Wenn keine Diskette im Laufwerk ist, wird das Bootprogramm aus dem Bootsektor (Master Boot Record) der ersten Festplatte ausgeführt. Es gibt verschiedene solcher Bootprogramme. Einige dieser Programme werten die Boot-Flags aus der Partitionstabelle aus, um zu bestimmen, von welcher Partition das Betriebssystem geladen werden soll.

Das Boot-Flag ist ein einzelnes Bit für jede Partition. Ist dieses Bit gesetzt, kann von dieser Partition ein Betriebssystem geladen werden.

Das MS-DOS Bootprogramm erwartet genau ein gesetztes Boot-Flag. LILO als primärer Bootloader ignoriert die Bootflags. Andere Bootmanager erlauben mehrere gesetzte Bootflags.

## Die Bedeutung des Partitionstyps

Um den verschiedenen Betriebssystemen die Möglichkeit zu geben, Partitionen anderer Formate zu erkennen, werden die Partitionen mit Kennzahlen (ID) versehen. Linux selbst kümmert sich nicht wirklich um die Partitionskenung, schließlich kann Linux mit einer Vielzahl fremder Dateisysteme gut umgehen. Der Kernel prüft beim Mounten eines Dateisystems immer die für das jeweilige Dateisystem typische magische Zahl und erkennt so, ob es sich um ein gültiges Dateisystem eines bestimmten Typs handelt.

Andere Betriebssysteme sind da viel weniger flexibel. In der Regel ordnet jeder Hersteller seinem Betriebssystem eine bestimmte Kennzahl für den Partitionstyp zu, mit der das Betriebssystem ``seine" Partitionen markieren und erkennen kann. *fdisk* kann z.Zt. 30 Kennzahlen bestimmten Betriebssystemen zuordnen. Es ist aber auch - mit ein paar Ausnahmen - möglich, beliebige andere ID's zu setzen.

Es ist nicht möglich, eine Partition vom oder nach dem Typ ``extended" (5) zu ändern. Es ist nicht möglich, einer leeren Partition (Typ 0) irgendeinen anderen Typ zu geben. Einer Partition den Typ 0 zuzuordnen, heißt sie zu löschen.

Die Typ-ID wird als hexadezimale Zahl angegeben. Die folgenden Typen werden erkannt:

0	Empty	9	AIX bootable	75	PC/IX	b7	BSDI fs
1	DOS 12-bit FAT	a	OS/2 Boot Manag	80	Old MINIX	b8	BSDI swap
2	XENIX root	40	Venix 80286	81	Linux/MINIX	c7	Syrinx
3	XENIX usr	51	Novell?	82	Linux swap	db	CP/M
4	DOS 16-bit <32M	52	Microport	83	Linux native	e1	DOS access
5	Extended	63	GNU HURD	93	Amoeba	e3	DOS R/O
6	DOS 16-bit >=32	64	Novell Netware	94	Amoeba BBT	f2	DOS secondary
7	OS/2 HPFS	65	Novell Netware	a5	BSD/386	ff	BBT
			8	AIX			

Voreinstellung für neue Linux-Partitionen ist `83' für das Linux-Dateisystem. Diese Kennzahl ermöglicht es den anderen aufgeführten Betriebssystemen, die Linux-Partitionen als ``fremd" zu erkennen.

Die ID 83 führt beim DR-DOS zu Problemen. Wenn eine Festplatte von Linux und DR-DOS gemeinsam genutzt werden soll, ist es sinnvoll, anstelle der 81 bzw. 83 als Partitionskennziffer die 42 oder 43 als Linux-ID zu verwenden.

## Das Dateisystem einrichten

Das Dateisystem präsentiert sich dem Benutzer als ein Verzeichnisbaum mit einer Menge verschiedener Dateien. Um diese Daten in der gewohnten Form zu organisieren, benötigt das Betriebssystem eine Grundstruktur auf dem Datenträger, in der es die Dateien einordnen kann. Diese Grundstruktur wird ebenfalls als Dateisystem bezeichnet, wenngleich es unmittelbar nach seiner Erzeugung noch keine echten Dateien sondern nur das Potential zur Aufnahme derselben enthält.

Diese Doppelbelegung des Begriffes erfordert manchmal etwas Überlegung, um den Sinn aus dem Zusammenhang zu erkennen. In diesem Abschnitt geht es um die Grundstruktur, das ``kleine" Dateisystem auf einer Partition. Der nächste Abschnitt behandelt dann die Zusammensetzung mehrerer kleiner Dateisysteme zu dem großen Dateisystem.

Linux kann mit mehreren Dateisystemtypen umgehen, deren Grundstruktur jeweils unterschiedlich ist. In der Praxis hat sich für die Verwendung auf reinen Linux-Partitionen das sogenannte *Zweite Erweiterte Dateisystem*, Extended 2 (EXT2), durchgesetzt. Es zeichnet sich durch hohe Geschwindigkeit, Sicherheit und Freiraum für zukünftige Erweiterungen aus. Im Anhang dieses Buches finden Sie eine ausführliche Beschreibung der Arbeitsweise von Dateisystemen allgemein und einen Vergleich der für Linux-Partitionen relevanten Typen.

Bei der Installation eines neuen Linux-Systems wird von den Installationsprozeduren immer das EXT2-Dateisystem erzeugt. Mit dem Systemprogramm [mke2fs](#) kann das EXT2-Dateisystem auch unabhängig von einem Installationsprogramm auf jeder Festplattenpartition eingerichtet werden. Die von `mke2fs` erzeugte Grundstruktur ist zu Beginn immer leer. Es gehen durch die Einrichtung eines neuen Dateisystems unbedingt alle bisher auf einer Partition gespeicherten Daten verloren.

# Das Dateisystem zusammenbauen

Linux organisiert alle Dateien auf Festplatten, Disketten, CDs und verwandten Medien in einem einzigen hierarchischen Dateisystem. Egal ob die Verzeichnisse auf der Linux-Partition einer lokalen Festplatte, auf einer DOS-Partition oder auf einem NFS-Server beheimatet sind, aus der Sicht des Benutzers werden alle Dateien auf einheitliche Weise präsentiert.

Um dieses homogene Bild zu erhalten, muß die Systemverwalterin die auf unterschiedlichen Medien (Festplattenpartitionen, CDs, NFS-Server) befindlichen Daten in geeigneter Form zusammenführen. Dazu werden die einzelnen Verzeichnisbäume auf den Medien zu einem einzigen, großen Dateisystem zusammengesetzt. Das Werkzeug, mit dem das Dateisystem zusammengebaut werden kann, ist das Systemprogramm [mount](#), um es wieder auseinanderzunehmen dient das Programm `umount`.

Ein typisches Beispiel für ein Kommando zum Einhängen einer Partition in das Dateisystem zeigt die Benutzung von `mount`:

```
[01] # mount -t ext2 -o grpquota /dev/hdb1 /usr/src
[02] # _
```

Durch das Kommando wird die erste Partition der zweiten Festplatte am ersten IDE Controller, `/dev/hdb1`, auf dem Verzeichnis `/usr/src` gemountet. Durch die Option `-t` wird der Typ des Dateisystems festgelegt, in diesem Fall `ext2`. Mit der Option `-o` wird dem Kerneltreiber beim Mounten des Dateisystems ein Schlüsselwort übergeben, in diesem Fall `grpquota`, das den Treiber veranlaßt, die Quotierung des Plattenplatzes auf der Basis von Gruppenzugehörigkeit abzurechnen.

## Die Dateisystemtypen von Linux 2.0

Mit der Kernelversion 2.0 unterstützt Linux 16 verschiedene Dateisystemtypen.

`ext2`

ist das zweite erweiterte Dateisystem von Remy Card. Es kann als das Standarddateisystem für neuere Linux-Installationen bezeichnet werden. Die Dateinamen können bis zu 255 Zeichen lang sein, die maximale Größe für eine Partition liegt bei 2 GB.

`iso9660`

ist das Dateisystem auf CD-ROMs. Der Linux-Kernel versteht die „Rock Ridge Extensions“ zum Standard ISO9660-Dateisystem, mit denen beispielsweise lange Dateinamen benutzt werden können. CD-ROMs müssen „Read-Only“ gemountet werden.

`proc`

ist das Prozeßdateisystem. Es handelt sich hierbei nicht um ein „physisches“ Dateisystem, sondern um eine Repräsentation relevanter Daten aus dem laufenden Linux-Kernel. Dieses Dateisystem wird üblicherweise auf dem Verzeichnis `/proc` aufgesetzt.

`nfs`

ist das Network File System. In diesem Dateisystem können Verzeichnisäste verschiedener Rechner im lokalen Netzwerk ausgetauscht werden.

smb

ist das *Samba Filesystem*, mit dem auf die exportierten Verzeichnisse eines Rechners mit NetBIOS- und TCP/IP-Unterstützung zugegriffen werden kann. Zum Beispiel bietet Windows für Workgroups diese Funktionalität, wenn ein TCP/IP-Stack installiert ist.

ncp

ist eine eingeschränkte Variante des NetWare Dateisystems. Es ist möglich, mit dem NCP-Protokoll auf Daten eines NetWare-Servers zuzugreifen.

ext

ist das alte erweiterte Dateisystem von Remy Card. Es ist durch das zweite erweiterte Dateisystem `ext2` abgelöst worden und ist nur noch auf alten Linux-Installationen zu finden.

xiafs

ist das alternative Linux-Dateisystem von Frank Q. Xia. Es ist „schlanker“ als das `ext2`, bietet aber weniger Features und Erweiterungsmöglichkeiten.

minix

ist das von A. Tanenbaum's Minix Betriebssystem übernommene Dateisystem. Es wird auch heute noch häufig für Diskettendateisysteme benutzt. Die Partitionsgröße ist auf maximal 64 MB beschränkt. Die Dateinamen können 14 (30) Zeichen lang sein.


msdos

ist das DOS Dateisystem . Linux kann DOS Disketten oder Festplattenpartitionen fest in den Verzeichnisbaum einbinden.

umsdos

ist eine zweite Ebene für DOS Dateisysteme , in der zusätzliche Features wie Permissions, lange Dateinamen, Links usw. realisiert werden können.

vfat

ist die erweiterte Variante des DOS-Dateisystems , die von Windows95 unterstützt wird. Hauptunterschied ist die Vergrößerung des Name-Space: es sind Dateinamen bis zu einer Länge von 255 Zeichen mit Unterscheidung von Groß- und Kleinschreibung erlaubt.

hpfs

ist das Dateisystem von OS/2. Dieses Dateisystem kann zur Zeit nur zum Lesen gemountet werden.

sysv

ist ein Dateisystem von Unix System V Release 2. Dieses Dateisystem wird zur Zeit nur auf Disketten unterstützt, Festplattenpartitionen können nicht gemountet werden. Mit der gleichen Option kann auch ein Xenix- oder Coherent-Diskettendateisystem gemountet werden.

affs

ist das *Amiga Fast Filesystem* mit seinen Varianten. Außer den speziell für Disketten optimierten Subtypen mit Directory-Cache, die nur zum Lesen geöffnet werden können, kann Linux im AFFS auch schreiben.

ufs

ist das Dateisystem der BSD-Systeme. Linux unterstützt zur Zeit nur das Lesen von Disketten und Festplattenpartitionen dieses Typs.

# Allgemeine Optionen beim Mounten der Dateisysteme

Ähnlich wie die Gerätetreiber, die durch Bootoptionen beeinflusst werden können, sind auch bestimmte Eigenschaften der Dateisysteme durch zusätzliche Optionen beim Mounten einer Partition oder eines Gerätes veränderbar.

Wie in dem Beispiel oben gezeigt, werden diese Optionen als Argument mit dem Schalter `-o` auf der Kommandozeile von `mount` angegeben. Es können mehrere Optionen durch Kommata getrennt zu einer Liste zusammengefügt werden. Die meisten Optionen können auch als feste Vorgabe für ein Dateisystem in der Datei `/etc/fstab` eingetragen werden. Zur Vereinfachung der Darstellung werden in dieser Liste auch Optionen aufgeführt, die entweder nur bei der interaktiven Benutzung von `mount` oder nur als Vorgabe in der `fstab` sinnvoll sind.

Einige der optionalen Veränderungen der Dateisystemparameter betreffen allgemeine Eigenschaften, die bei jedem der unterstützten Typen zu finden sind. Diese Optionen können also beim Mounten jedes Dateisystems angegeben werden.

`defaults`

entspricht normalerweise der Kombination `suid, rw`. Wenn ein Dateisystem von jedem Systembenutzer eingebunden werden kann (mit der Option `user, s.u.`), entspricht `default` für nicht privilegierte Benutzer der Kombination `nosuid, noexec, nodev, rw`.

`noauto`

(nur `fstab`) verhindert das automatische Einbinden der Partition.

`noexec`

Das Betriebssystem ignoriert die Executable-Bits im Permission-Feld, verbietet also die Ausführung jedes Programms von dieser Partition.

`nosuid`

unterdrückt die Wirkung der SUID und SGID Bits bei der Ausführung von Programmen aus dieser Partition.

`nouser`

(Voreinstellung, nur `fstab`) verbietet ausdrücklich die Benutzung von `mount` durch normale Systembenutzer.

`nodev`

die zeichen- und blockorientierten Gerätedateien in dieser Partition werden nicht angesprochen.

`rw`

erlaubt das Lesen und das Schreiben auf dieser Partition.

`ro`

verbietet das Schreiben auf diese Partition.

`sync`

die Metadaten (Superblock, Inode, Verzeichnisdaten) werden ungepuffert (synchron) auf das Speichermedium geschrieben (nur beim `ext2fs` implementiert).

`user`

(nur `fstab`) Mit dieser Option kann die Superuserin das Einbinden von Dateisystemen durch



normale Systembenutzer erlauben. Die Benutzer können dann die Kommandos `mount` und `umount` benutzen, denen sie entweder die Gerätedatei oder das Verzeichnis zum Aufsetzen als Parameter übergeben können.

`remount`

(nur `mount`) Diese Option veranlaßt `mount` ein bereits aufgesetztes Dateisystem mit veränderten Optionen neu zu Mounten. Diese Option wird benutzt, um beim Systemstart nach dem Filesystemcheck die Schreibberechtigung für das read-only gemountete Rootfilesystem zu erteilen. Dateisystemspezifische Einstellungen können mit `remount` nicht geändert werden.

## Gemeinsame Optionen für verschiedene Dateisystemtypen

Einige Dateisysteme haben spezielle Features, die durch zusätzliche Optionen beeinflusst werden können. In diesem Abschnitt werden zunächst Optionen beschrieben, die von mehreren Dateisystemtypen unterstützt werden. In den darauffolgenden Abschnitten werden jeweils die speziellen Optionen für die einzelnen Typen behandelt.

`conv=Modus`

(für FAT, hpfs und iso9660 Dateisysteme) schaltet die automatische Konvertierung von Zeilenenden (`LINEFEED` und `CARRIAGE RETURN`) von dem Dateisystem zu. Der *Modus* bestimmt, welche Daten konvertiert werden.

`binary`

(default) Es findet keine Konvertierung statt.

`text`

Beim Lesen werden `CR/LF` zu `LF`, nur bei FAT wird beim Schreiben `LF` zu `CR/LF`.

`auto`

Dateien mit bekannten Endungen für Binärdateien (`exe`, `com`, `dll`, `lzh`, `tif` etc.) werden nicht konvertiert.

`mtext`

(nur bei iso9660) Wie `text`, es werden `CR` zu `LF` konvertiert.

`debug`

(nur ext2 und FAT) Es werden die Dateisystemparameter ausgegeben.

`gid=Wert`

bei FAT und hpfs Dateisystemen wird allen Dateien die angegebene Gruppen-ID zugeordnet.

`uid=Wert`

bei FAT und hpfs Dateisystemen wird allen Dateien die angegebene User-ID zugeordnet.

`umask=Wert`

läßt die Zugriffsrechte für Dateien und Verzeichnisse im FAT oder hpfs Dateisystem durch die Maske *Wert* erscheinen. Der Wert wird als Oktalzahl eingegeben und interpretiert wie beim Shellkommando [`umask`](#) beschrieben.

## Spezielle Optionen für das EXT2FS

`check[=Art]`

veranlaßt den Kernel, beim Mounten eines ext2fs eine Konsistenzprüfung durchzuführen. Die *Art* der Prüfung kann optional angegeben werden. Dabei bedeutet:

none

Es wird keine Prüfung durchgeführt. Diese Option kann auch mit `nocheck` abgekürzt werden.

normal

(default) Es werden die Inodes und die Bitmaps geprüft.

strict

Es wird zusätzlich geprüft, ob die freien Blöcke im Datenbereich liegen.

Wenn der Kernel einen Fehler feststellt, wird das Valid-Flag auf einen speziellen Wert gesetzt, um dem `e2fsck`-Programm den Fehler anzuzeigen (nur für ext2fs). Zusätzlich kann durch die `errors` Option das Verhalten des Kernels nach Entdeckung eines Fehlers eingestellt werden.

`errors=`**Aktion**

Diese Option regelt das Verhalten vom ext2 Dateisystem nach der Erkennung von Fehlern.

continue

Das Dateisystem wird ohne weitere Aktion als fehlerhaft markiert.

remount-ro

Das Dateisystem wird in den ``read only" Modus gebracht.

panic

Eine Kernelpanik wird erzeugt (das Dateisystem wird synchronisiert und der Kernel angehalten).

`sysvgroups` | `nogrp`

veranlaßt das ext2fs, Dateien nur dann mit der Gruppen-ID des Verzeichnisses zu erzeugen, wenn das SGID-Bit des Verzeichnisses gesetzt ist. Sonst wird die Datei (wie bei allen anderen Linux-Dateisystemen) mit der GID des erzeugenden Prozesses angelegt.

`sb=`**Zahl**

veranlaßt den Kernel, den Superblock aus dem mit der Zahl bezeichneten Block einer ext2fs Partition zu lesen. Das ext2fs legt typischerweise Kopien des Superblocks in den Blöcken 8193, 16385 usw. an (nur für ext2fs).

`minixdf`

verändert den `statfs` Aufruf des ext2fs so, daß er die Gesamtzahl der Datenblöcke inklusive der vom Dateisystem selbst belegten Blöcke liefert.

`bsddf`

Der `statfs` Systemaufruf liefert als Gesamtgröße des Dateisystems den maximalen verfügbaren Platz, das ist die Differenz aus der physikalischen Größe und dem vom Dateisystem selbst bereits belegten Platz.

`bsdgrps` | `grp`

veranlaßt das ext2fs, Dateien mit der Gruppen-ID des Verzeichnisses zu erzeugen (wie bei BSD).

`resuid=`**UID**

`resgid=`**GID**

Mit diesen beiden Optionen wird festgelegt, welche User bzw. Benutzergruppe den reservierten

Datenbereich einer EXT2-Partition benutzen darf. Die Voreinstellung des Systems ist für beide Werte die ID 0, der Bereich ist also für den Superuser reserviert.

quota | usrquota | grpquota

Die Quota-Optionen sind erlaubt aber wirkungslos: sie werden ignoriert.

## Spezielle Optionen für die FAT-Dateisysteme

Unter dem Sammelbegriff FAT-Dateisysteme werden die von DOS, Windows95 und Windows-NT benutzten Dateisysteme mit *File Allocation Table* zusammengefaßt. Linux unterstützt diese Filesysteme in verschiedenen Variationen mit den Typen `msdos`, `umsdos` und `vfat`.

Wegen der gemeinsamen Grundstruktur teilen sich diese drei Typen folgende Optionen:

`check=Art`

Durch diese Einstellung wird festgelegt, wie streng die DOS Namenskonvention beim Erzeugen von Dateien geprüft wird:

`relaxed`

Lange Dateinamen werden gekürzt, Leer- und Sonderzeichen (`?<*=` etc.) sind erlaubt.

`normal`

Wie `relaxed`, die meisten Sonderzeichen (`?<*`) sind nicht erlaubt.

`strict`

Lange Dateinamen werden nicht automatisch gekürzt, sind also schlicht verboten.

`fat=Wert`

erzwingt die Bearbeitung der FAT mit 12 oder 16 Bit Einträgen, unabhängig von der automatischen Erkennungsroutine.

`quiet`

unterdrückt beim DOS Dateisystem die Fehlermeldungen beim Versuch, Eigentümer oder Modus einer Datei zu verändern.

Außerdem werden die Optionen `debug`, `conv`, `gid`, `uid` und `umask` unterstützt, die auch für andere Dateisysteme benutzt werden können.

Speziell für den Dateisystemtyp `vfat` gibt es noch drei zusätzliche Optionen:

`uni_xlate`

Wenn diese Option gesetzt ist, werden Unicode-Zeichen in Dateinamen durch eine Sequenz aus Doppelpunkt und einem unter Linux darstellbaren Zeichen umgewandelt.

`posix`

Diese Option ermöglicht die Unterscheidung von Groß- und Kleinschreibung bei Dateinamen.

`nonumtail`

Mit dieser Option kann die Methode zur Erzeugung von 8.3 Dateinamen verändert werden. Normalerweise wird der Basisteil eines Dateinamens beim Kürzen mit einer Seriennummer versehen. Wenn diese Option gesetzt ist, werden Dateinamen, die bei der Kürzung auf 8 Zeichen eindeutig sind, ohne Seriennummer gekürzt.

# Spezielle Optionen für das iso9660 Dateisystem

block=**Größe**

setzt die Blockgröße bei iso9660 Dateisystemen auf den angegebenen Wert. Mögliche Werte sind 512, 1024 und 2048, Voreinstellung ist 1024.

cruft

Diese Option kann beim Mounten defekter CD-ROMs hilfreich sein.

map=**Art**

Bei iso9660 Dateisystemen ohne Rock-Ridge Erweiterung bestehen Dateinamen aus Großbuchstaben und werden von der Zeichenkette `;1' abgeschlossen. Normalerweise (normal) wird die Endung abgeschnitten und der Name in Kleinbuchstaben umgesetzt. Dieses Mapping kann auch aus (off) geschaltet werden. Bei Dateisystemen mit Rock-Ridge Erweiterung muß der norock Schalter gesetzt sein, damit die map Option wirksam wird.

norock

schaltet die Rock-Ridge Erweiterung bei iso9660 Dateisystemen ab.

mode=**Rechte**

stellt die Zugriffsrechte für Dateien auf CDs ohne Rock-Ridge Erweiterung ein. Vorgegeben ist freies Leserecht für alle User.

unhide

läßt normalerweise unsichtbare Dateien wieder auftauchen.

## Spezielle Optionen für Netzwerk-Dateisysteme (NFS und SMB)

Die mit den Linux-Distributionen zur Zeit ausgelieferte Version von mount unterstützt nur das NFS. Für Samba gibt es ein spezielles Programm, smbmount.

Beide Varianten verstehen eine Vielzahl von Argumenten, aus denen sie eine spezielle Datenstruktur generieren, die sie dem Treiber für das entsprechende Dateisystem übergeben. In der folgenden Liste werden die wichtigsten dieser Optionen aufgeführt, eine vollständige Erklärung aller Einstellungen finden Sie in den Manualpages zu nfs(5).

rsize=**Bytes**

stellt die Größe eines Datenpakets beim Lesen vom Server ein. Vorgabe des Kernels ist 1024 Bytes.

wsize=**Bytes**

stellt die Größe eines Datenpakets beim Schreiben auf einen Server ein. Vorgabe des Kernels ist 1024 Bytes.

timeo=**Zehntelsekunden**

setzt die Zeit, die nach einem RPC-Timeout vergeht, bevor ein Datenpaket zum zweiten Mal gesendet wird. Voreinstellung sind 7 Zehntel Sekunden. Nach der ersten Retransmission wird die Zeitspanne verdoppelt, bis das Maximum von 60 Sekunden erreicht ist. Danach wird die Kaskade mit verdoppeltem Startwert erneut durchlaufen.

retrans=**Anzahl**

stellt die maximale Anzahl der Durchläufe der oben beschriebenen Timeout-Kaskade ein. Wenn danach keine Verbindung mit dem Server besteht, wird die Fehlermeldung ``server not responding" ausgegeben.

soft

Wenn dieser Schalter gesetzt ist, wird das mount-Kommando abgebrochen, nachdem die mit retrans bestimmte Anzahl von Timeout-Kaskaden durchlaufen wurde.

hard

Beim harten Mounten wird ohne Ende versucht, eine Verbindung zum Server herzustellen. Das ist die vorgegebene Methode.

intr

Wenn dieser Schalter gesetzt ist, können Dateioperationen auf dem Netzwerk-Dateisystem durch ein Signal unterbrochen werden. Normalerweise sind Unterbrechungen nicht erlaubt.

## Feste Vorgaben für die Zusammensetzung des Dateisystems

Um bestimmten Festplattenpartitionen oder Geräten Dateisystemtypen und Mountoptionen fest zuzuordnen, können diese Parameter in der Datei `/etc/fstab` verzeichnet werden. Das Systemprogramm `mount` benutzt diese Datei, um das Dateisystem beim Systemstart automatisch zusammenzusetzen und um bei der interaktiven Benutzung fehlende Parameter zu ergänzen.

Die Einträge haben folgende Form:

*Device Mountpunkt Typ Optionen Dump Check*

Die sechs Felder werden durch Leerzeichen voneinander getrennt, Kommentare werden durch das Nummernzeichen `#` eingeleitet.

### Device

Im ersten Feld wird die Gerätedatei angegeben, für die der Eintrag gelten soll. Das kann eine Festplattenpartition sein, aber auch die Gerätedatei für ein CD-ROM- oder Diskettenlaufwerk. Der Name wird mit absolutem Pfad eingetragen, also zum Beispiel `/dev/hda1` für die erste Festplattenpartition, `/dev/fd0` für die Floppy usw.

### Mountpunkt

Im zweiten Feld wird angegeben, auf welchem Verzeichnis die Partition gemountet werden soll. Wenn die Partitionen automatisch zusammengesetzt werden sollen, muß durch die Reihenfolge der Einträge in der `fstab` sichergestellt werden, daß das Verzeichnis auch tatsächlich im Dateisystem vorhanden ist.

### Typ

Das dritte Feld ordnet der Partition oder dem Gerät einen Dateisystemtyp zu. Die Liste der von Linux unterstützten Dateisysteme finden Sie am Anfang dieses Abschnitts.

### Optionen

Im vierten Feld werden die Optionen eingetragen, mit denen das Dateisystem gemountet werden soll. Es sind alle Optionen erlaubt, die auch bei der interaktiven Benutzung von `mount` akzeptiert werden. Das Feld kann eine Liste von mehreren, durch Kommata getrennten Optionen enthalten.

### Dump

Das fünfte Feld dient zur Markierung eines Dateisystems als Kandidat für die Datensicherung durch das Systemprogramm `dump`. Mit `dump` können nur EXT2 Dateisysteme gesichert werden.

## Check

Mit dem sechsten Feld kann die Reihenfolge bestimmt werden, in der die Dateisysteme beim automatischen Filesystemcheck während des Bootens geprüft werden. Das Rootfilesystem sollte hier als einziges mit der 1 gekennzeichnet werden, damit dessen Prüfung als erstes und allein erfolgt. Partitionen auf unterschiedlichen Geräten können danach parallel geprüft werden. Die Reihenfolge der Prüfung von Partitionen auf der gleichen Festplatte können Sie durch fortlaufende Nummern selbst bestimmen. Partitionen, die im sechsten Feld mit einer Null gekennzeichnet sind, werden nicht automatisch einem Filesystemcheck unterworfen.

# Die Konsistenz des Dateisystems prüfen

Wenn der Rechner unkontrolliert ausgeschaltet wurde, oder wenn es zu einem Systemabsturz gekommen ist, muß damit gerechnet werden, daß sich das Dateisystem in einem inkonsistenten Zustand befindet.

Fehler im Dateisystem können auftreten, weil die Schreiboperationen durch den Festplattencache im Arbeitsspeicher gepuffert werden und diese Daten beim Abschalten des Rechners verloren gehen. Andere Fehler entstehen, wenn das System mitten in einer ungepufferten Schreiboperation seinen Dienst aufgibt.

Anlegen, Erweitern und Löschen von Dateien sind keine einfachen Abläufe, die in einem "atomaren" Schritt durchgeführt werden können. Allokierung einer I-Node, Eintrag des Dateinamens, Allokierung der Datenblöcke und Beschreiben der Blöcke mit Daten sind einzelne Abschnitte des Prozesses, die mit mehr oder weniger langen Unterbrechungen nacheinander ausgeführt werden, um das Ziel zu erreichen.


Die meisten bei einem Systemabsturz entstehenden Fehler betreffen die Daten in den Dateien, die bis zum Abschalten des Rechners nicht ordentlich geschlossen wurden. Die Reparatur dieser Daten kann nur von den Programmen oder Usern vorgenommen werden, die diese Dateien angelegt haben.

Es können aber auch Fehler in der Struktur des Dateisystems entstehen, die durch geeignete Programme erkannt und repariert werden können. Bei dem am häufigsten benutzten Dateisystem für Linux, dem `ext2fs`, können beispielsweise folgende Fehler behoben werden:

- fehlerhafte Einträge in Verzeichnissen,
- fehlerhafte Einträge in I-Nodes,
- Dateien, die in keinem Verzeichnis eingetragen sind und
- Datenblöcke, die zu mehreren verschiedenen Dateien gehören.

Nicht alle Reparaturen können ganz ohne Datenverlust vorgenommen werden, das Dateisystem kann aber in der Regel wieder in einen konsistenten Zustand gebracht werden. Wenn beim File-System-Check eine Datei gefunden wird, die in keinem Verzeichnis eingetragen ist, wird sie automatisch in das Verzeichnis `./lost+found` mit der Nummer der I-Node eingetragen.

Wenn das System nicht korrekt heruntergefahren wurde, erkennt das System das beim nächsten Booten am Status des Ext2-Dateisystems. Bei einem ordentlichen Systemabschluß werden die Dateisysteme abgebaut und dabei das `Valid-Flag` im Superblock gesetzt. Beim Booten kann diese Information aus dem Superblock genutzt werden, um die möglicherweise fehlerhaften Dateisysteme automatisch während der Systeminitialisierung testen und nötigenfalls reparieren zu lassen.

Bei allen Linux-Distributionen enthalten die Shellscripts  zur Systeminitialisierung, die von dem Programm `init` nach dem Booten ausgeführt werden, die notwendigen Kommandos zur Durchführung eines File-System-Check.

Jedes der unter Linux verwendeten Dateisysteme benötigt sein spezielles Programm für den File-System-Check. Zu diesem Zweck existieren die Systemprogramme `fsck.minix`, `fsck.ext2` und `fsck.xiafs`. Das Programm [fsck](#) arbeitet als Front-End für alle File-System-Checker und ermöglicht es, mit einem einzigen Aufruf alle Dateisysteme gemeinsam zu prüfen.

Wenn Sie die Konsistenz eines Dateisystems prüfen wollen, müssen Sie damit nicht bis zum nächsten Booten warten. Sie können die Programme zum File-System-Check zu jedem beliebigen Zeitpunkt aufrufen. Sollte ein Dateisystem allerdings Fehler enthalten, darf die Reparatur nur auf einem Dateisystem durchgeführt werden, auf das der Kernel keinen Schreibzugriff hat. Diese Einschränkung ist notwendig, damit sich Kernel und Reparaturprogramm bei der Veränderung des Dateisystems nicht in die Quere kommen. Auch aus diesem Grund bietet sich der automatische Test vor dem Zusammenbau des Dateisystems beim Booten an.

## Das Rootfilesystem reparieren

Der Linux-Kernel kann das Rootfilesystem beim Booten im Read-Only mounten. Dazu muß ein entsprechendes Flag in der Kerneldatei gesetzt sein (→ [rdev](#)) oder es muß die entsprechende Option vom LILO-Bootloader übergeben werden.

Wenn Sie das System im Single-User-Mode starten, können Sie die Schreibberechtigung für das Rootfilesystem auch "von Hand" ändern. Das Systemprogramm [mount](#) ermöglicht dieses remount zum Ändern der Optionen.

Die Kommandos zum Wechseln des Schreib-Lese-Status sehen folgendermaßen aus:

```
# mount -n -o remount,ro /
# touch /foo
touch: foo: Permission denied
# mount -n -o remount,rw /
# _
```

Um das Kommando zum Löschen der Schreibberechtigung ausführen zu können, darf kein Prozeß eine Datei im Rootfilesystem zum Schreiben geöffnet haben. Sie müssen also, notfalls mit `kill`, alle Programme beenden, die noch offene Dateien haben. In der Regel sind das Dämonen wie `syslogd`, `lpd` oder `crond`. Beim `kmem-ps` gibt es das Programm `fstat`, das dabei helfen kann, die problematischen Prozesse zu identifizieren.

Auch wenn das Rootfilesystem Read-Only gemountet ist, bleibt ein Teufelskreis bei dessen Reparatur erhalten: die Programme zum Reparieren werden von dem eventuell defekten Dateisystem geladen.

Um dieses Problem zu umgehen, können Sie ein Minimalsystem von Diskette laden, wie es als

``*Rescue-System*'' von einigen Linux-Distributionen angeboten wird. Nachdem das System mit einer „bootable Rootdisk`` gestartet ist, kann dann das Rootfilesystem auf der Festplatte von außen repariert werden.

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [Softwaremanagement](#) **Up:** [Systemverwaltung](#) **Previous:** [Benutzer eintragen](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)



## Subsections

- [Vom Sourcepaket zum fertigen Programm](#)
  - [Paketmanagement mit RPM](#)
    - [Installation und Upgrade einer Software](#)
    - [Untersuchung von RPM-Paketen](#)
    - [Direkter Zugriff auf die Daten in der RPM-Datei](#)
    - [RPM und die Konfiguration der Software](#)
- 

# Softwaremanagement

Ein Linux-System ist kein statisches Gebilde, das einmal installiert und dann nicht mehr verändert werden kann. Die große Vielfalt Freier Software und die sehr unterschiedlichen Einsatzgebiete von Linux machen Linux im Gegenteil zum flexibelsten Betriebssystem auf dem Markt. Bereits bei der Installation läßt sich die Zusammenstellung des Systems bis hinunter zur Auswahl einzelner Programme individuell anpassen.

Nach Abschluß der Erstinstallation gibt es verschiedene Möglichkeiten, neue Software zum System hinzuzufügen oder installierte Pakete wieder zu entfernen. Die modernen Linux-Distributionen kommen alle mit komfortablen Tools zur Verwaltung der Softwarepakete. In den meisten Fällen läßt sich damit das Softwaremanagement über eine einfache Menüsteuerung oder unter der grafischen Oberfläche mit der Maus durchführen.

Die Objekte, mit denen das Softwaremanagement unter Linux arbeitet, sind sogenannte Pakete. In einem Paket sind mehrere Dateien zusammengepackt. Gemeinsam mit den Daten ist mindestens die Information, in welchem Verzeichnis diese Dateien installiert werden sollen, in dem Paket enthalten. Je nach Art und Herkunft des Paketes können zusätzlich Informationen über die Abhängigkeit dieses Pakets von anderen Paketen, Anweisungen für die automatische Vorbereitung der Installation und verschiedene ähnliche Metadaten Bestandteile eines Pakets sein.

Die Pakete lassen sich nach Herkunft, Form und Inhalt in verschiedene Kategorien aufteilen:

## Sourcepakete

enthalten den Quelltext für das Programm und sämtliche dazugehörenden Dateien. Jedes Sourcepaket läßt sich auf eine Release (Freigabe) von dem oder den Entwickler(n) zurückführen.

Die Freigabe von Quelltexten für Computerprogramme hat eine lange Tradition in der UNIX-Welt. Indem Linux die gleiche Betriebssystemschnittstelle wie UNIX benutzt, können

praktisch alle für UNIX entwickelten Programme auch unter Linux verwendet werden. Fast die gesamte Software, die heute mit einer Linux-Distribution ausgeliefert wird, hat ihren Ursprung in einem Sourcepaket für das traditionelle UNIX.

Ob es sich bei einer Datei um ein Sourcepaket handelt, ist von außen in der Regel nicht festzustellen. Manchmal enthält der Dateiname eines Sourcepakets den Abschnitt `.src..`. Möglicherweise ist auch das Gegenstück zum Sourcepaket, nämlich das Binärpaket, durch den Abschnitt `.bin.` gekennzeichnet, was automatisch die ansonsten gleichnamige Datei als Sourcepaket kennzeichnet. Aufgrund der UNIX-Tradition ist es sehr wahrscheinlich, daß TAR-Archive, die nicht zu einer Linux-Distribution gehören, ein Sourcepaket enthalten.

Nicht jedes Sourcepaket von UNIX läßt sich ganz ohne Anpassung für ein Linux-System verwenden. Aus der Dynamik der frühen Linux-Entwicklung heraus werden viele Sourcepakete in einer speziellen Linux-Versionen neu gepackt und über bestimmte FTP-Server verteilt (`tsx-11.mit.edu` und `sunsite.unc.edu`).

Mit den Linux-Distributionen werden, der GNU General Public License entsprechend, ebenfalls Sourcepakete zu allen Freien Programmen ausgeliefert. Die in diesen Paketen enthaltenen Sourcen können wiederum spezifische Veränderungen des jeweiligen Distributors enthalten. Solche Änderungen sind entweder unsichtbar in den Programmtext integriert oder (zusätzlich) als DIFF-Datei dem Paket beigelegt. Diese Sourcepakete werden bei der Installation nicht auf die Festplatte kopiert. Sie befinden sich in der Regel in einem separaten Ast des Verzeichnisbaums auf der CD.

Um aus einem Sourcepaket ein ausführbares Programm zu gewinnen, muß es in der Regel in einem Seitenast des Verzeichnisbaums auf der Festplatte ausgepackt, mit Hilfe eines Compilers übersetzt und die dabei entstehenden Dateien anschließend installiert werden.

## **Binärpakete**

enthalten ausführbare Programmdateien für eine bestimmte Rechnerarchitektur und für ein bestimmtes Betriebssystem, gemeinsam mit den zur Laufzeit benötigten Daten, Konfigurationsdateien und der Dokumentation.

Die Verbreitung von Linux-Software in Form von Binärpaketen hat sich mit den Linux-Distributionen etabliert. Die Geschichte der Distributionen reicht von MCC-Interim über SLS und Slackware zu der gegenwärtigen Vielfalt unterschiedlicher Distributionen auf dem Markt.

Gelegentlich sind Sourcepakete durch den Abschnitt `.bin.` im Dateinamen gekennzeichnet. Ohne diese Kennzeichnung sind Binärpakete fast ausschließlich bei den verschiedenen Linux-Distributionen zu finden.

Mit Ausnahme einiger weniger kommerziell lizenzierter Produkte stammen alle Binärpakete direkt von entsprechenden Sourcepaketen ab. Die Leistung der Distributoren besteht in erster Linie darin, die Software in ein konsistentes Dateisystemmodell einzubauen, für eine möglichst einfache Installation zu sorgen und eine sinnvolle Defaultkonfiguration vorzunehmen.

Durch die Verbreitung in fertigen Binärpaketen ist die Installation von Linux-Software heute ebenso einfach wie die der Produkte von Microsoft und Co. Die Programme werden durch ein einzelnes Installationskommando gebrauchsfertig eingerichtet. Selbst auf Spielereien mit

grafischer Oberfläche braucht die Systemverwalterin bei der Installation von Linux-Software heute nicht mehr zu verzichten.

## TAR-Archive

sind der Standard für die Verbreitung von Quellen für Freie Software im Internet. Der Name leitet sich von *Tape ARchive* ab und ist Konzept. TAR-Archive können direkt auf die unterschiedlichsten Medien geschrieben oder als Dateien in jedem Dateisystem angelegt werden. Der größte Vorteil dieses Archivformats besteht in seiner Portabilität. Eine Version des Programms `tar` zur Bearbeitung der TAR-Archive ist praktisch für jedes Betriebssystem erhältlich. TAR-Archive können auf jedem UNIX-System (natürlich auch unter Linux) erzeugt und gelesen werden.

In TAR-Archive werden Dateien und ganze Verzeichnisbäume eingepackt. In der Regel werden diese Archive auch in dem zweiten Sinn gepackt: durch ein Kompressionsprogramm (heute in der Regel `gzip`) werden die im Archiv zusammengefaßten Daten verdichtet.

TAR-Archive lassen sich leicht an der typischen Endung `.tar` erkennen. Je nachdem, ob sie mit `compress` oder mit `gzip` komprimiert sind, kann die Endung zu `.tar.Z` oder `.tar.gz` erweitert sein. Um Dateien im TAR-Format auch in einem DOS-Dateisystem speichern zu können, hat sich auch die Endung `.tgz` für mit `gzip` komprimierte TAR-Archive etabliert.

TAR-Archive werden mit dem Programm [`tar`](#) bearbeitet.

Zu Beginn der Entwicklungsgeschichte der Linux-Distributionen war das TAR-Format auch der Standard für die Verteilung der fertig übersetzten, binären Software. Als Pakete waren diese Archivdateien aber nur begrenzt geeignet. Es hat sozusagen an der Verpackung gefehlt. TAR-Archive haben kein reguläres Feld für Absendervermerke.

Durch die Entwicklung spezieller Paketformate, die den Anforderungen bezüglich Softwaremanagement viel besser gerecht werden, gerät das TAR-Archiv bei der Distribution von gebrauchsfertigen Softwarepaketen mehr und mehr in den Hintergrund.

## RPM-Pakete

sind speziell für die Distribution und das damit verbundene Softwaremanagement entworfene Softwarepakete. Das Format wurde zunächst für die RedHat-Distribution entwickelt, wird aber heute von weiteren Distributoren benutzt.

Die Dateinamen der RPM-Pakete tragen typischerweise die Endung `.rpm`.

Als eigentliche "Fracht" ist in jedem RPM-Paket ein CPIO-Archiv eingepackt. In der Funktion sind CPIO-Archive den TAR-Archiven gleichwertig, sie werden jedoch mit dem Programm [`cpio`](#) erzeugt und verwaltet.

Mit dem CPIO-Archiv selbst hat die Systemverwalterin bei der Benutzung von RPM-Paketen in der Regel nichts zu tun. Das Archiv kann zwar unkompliziert mit Hilfe des Programms `rpm2cpio` extrahiert werden, der *RedHat Packet Manager* [`rpm`](#) übernimmt das Handling des Archivs aber vollständig und erledigt darüberhinaus wesentliche Aufgaben des Softwaremanagements.

Im Unterschied zu `tar` gehören `rpm` und `rpm2cpio` nicht zu den Standardutilities und sind deshalb nicht automatisch in jeder Linux-Distribution enthalten. Die Nachrüstung ist nicht schwieriger als die Installation irgendeiner anderen Software. Allerdings kann ein nachträglich installiertes RPM-System nur die Software verwalten, die mit diesem System neu installiert

wurde.

## DEB-Pakete

sind speziell für die Debian-Distribution gepackt. Der Aufbau dieser Pakete ist ebenfalls speziell für die Anforderungen von Distribution und Softwaremanagement entwickelt worden.


Auch Debian-Pakete können leicht an ihren Dateinamen erkannt werden: sie tragen die Endung `.deb`.

Intern sind DEB-Pakete als verschachteltes Archiv aufgebaut: die DEB-Datei ist ein AR-Archiv, darin enthalten sind zwei komprimierte TAR-Archive und eine Indikatordatei. Zum Auspacken eines DEB-Pakets sind deshalb im Notfall nur die Standardtools `ar` und `tar` erforderlich. Die volle Leistungsfähigkeit des DEB-Formats erschließt sich aber erst durch die Verwendung der Debian Pakettools `dpkg` und `dselect`.

Mit den modernen Linux-Distributionen ist die Installation eines Systems mit vielen hundert lauffähigen Programmen ein Kinderspiel. Tools wie `rpm` oder `dselect` nehmen der Systemverwalterin viel Arbeit beim Hinzufügen oder beim Entfernen von Softwarepaketen ab. Linux hat so innerhalb weniger Jahre den Ruf eines Hacker-Betriebssystems abgelegt und steigt stetig in der Gunst "normaler" Benutzer. Das ist sehr gut so und diese Entwicklung ist noch nicht abgeschlossen. Der schlechte Ruf von UNIX, das als kompliziert und schwer bedienbar gilt, wird von Linux nicht übernommen.

## Vom Sourcepaket zum fertigen Programm

Die Möglichkeit, Freie Software als Binärpaket ohne Aufwand in kürzester Zeit installieren zu können ist eine Bereicherung. Es ist aber sowohl für den einzelnen Anwender als auch für die Zukunft von Linux allgemein wichtig, daß die gleiche Software überall auch aus den Sourcen neu erzeugt werden kann. Nur so hat jeder Anwender die Möglichkeit, Anpassungen, Fehlerkorrekturen und Verbesserungen an seinem System unabhängig durchzuführen. Die massenhafte Verbreitung der Sourcen ermöglicht die Emanzipation der Benutzer und die aktive Gestaltung und Entwicklung neuer Software nach den eigenen Bedürfnissen.

Die Faszination von Freier Software entsteht so richtig erst beim Mitmachen. Es ist das reale Erlebnis von Freiheit  und Abenteuer: in den unermesslichen Tiefen des Internet gibt es ständig neue Software zu entdecken. Aus kleinen Keimen können, durch die gemeinsame Anstrengung vieler, große Verzeichnisbäume wachsen.

Come to where the future is

Come to Linux Country

Ende der Werbepause : - )

Es gehört kein Informatikstudium und schon gar keine Zauberkraft dazu, ein UNIX-Programm unter Linux zu übersetzen. Eine Voraussetzung muß allerdings erfüllt sein: das Entwicklungssystem muß installiert sein.

Das folgende Beispiel zeigt, wie der Midnight Commander (`mc`) aus den originalen Sourcen erzeugt wird.

Zuerst muß das Sourcepaket per FTP geholt werden. Dazu eignet sich jeder FTP-Client. Voraussetzung ist natürlich ein funktionierender Internetanschluß.

Ohne Internetanschluß können Sourcepakete auch von einer der vielen Archiv-CDs gezogen werden, die beispielsweise im Buchhandel vertrieben werden.

```
[01] $ cd /usr/src
[02] $ ncftp prep.ai.mit.edu:/pub/gnu/mc-4.1.tar.gz
...
Receiving file: mc-4.1.tar.gz
mc-4.1.tar.gz: 1093859 bytes received in 216.24 seconds, 4.94 K/s.
[03] $ ls -l mc-4.1.tar.gz
-rw-rw-r-- 1 she she 1093859 Sep 17 01:40 mc-4.1.tar.gz
[04] $ _
```

Bei dem Softwarepaket handelt es sich um ein mit `gzip` komprimiertes TAR-Archiv. Diese Datei wird mit dem Programm `tar` bearbeitet. Das Dekomprimieren kann `tar` "on the fly" erledigen, indem es mit der Option `-z` aufgerufen wird.

Vor dem Auspacken ist es immer sinnvoll, zuerst ein Listing des Inhalts anzusehen. Der Aufwand für diese kurze Überprüfung steht in keinem Verhältnis zu dem Ärger, den ein schlecht gepacktes TAR-Archiv machen kann. Es kann nämlich passieren, daß das Archiv ohne Verzeichnis erzeugt wurde und deshalb alle Dateien im aktuellen Verzeichnis ausgepackt werden!


```
[04] $ tar tvfz mc-4.1.tar.gz
drwxr-xr-x miguel/gnu 0 1997-09-16 23:55 mc-4.1/
drwxr-xr-x miguel/gnu 0 1997-09-16 23:54 mc-4.1/src/
-rw-r--r-- miguel/gnu 2588 1997-09-16 23:54 mc-4.1/src/color.h
-rw-r--r-- miguel/gnu 1265 1997-09-16 23:54 mc-4.1/src/file.h
...
-rw-r--r-- miguel/gnu 6 1997-09-16 23:55 mc-4.1/os2/os2edit/dirtools.h
-rw-r--r-- miguel/gnu 25471 1997-09-16 23:55 mc-4.1/os2/os2edit/makefile.de
bug
-rw-r--r-- miguel/gnu 24935 1997-09-16 23:55 mc-4.1/os2/os2edit/makefile.re
lease
-rw-r--r-- miguel/gnu 95 1997-09-16 23:55 mc-4.1/os2/os2edit/makefile.rf
[05] $ _
```

Das Listing zeigt, daß sämtliche Dateien aus dem TAR-Archiv in das Verzeichnis `mc-4.1` installiert werden. Dieses Verzeichnis wird beim Auspacken automatisch erzeugt.

```
[05] $ ^tv^x
tar xfz mc-4.1.tar.gz
[06] $ cd mc-4.1
[07] $ ls -F
COPYING      README.OS2*  doc/         os2/
FAQ          VERSION     edit/        slang/
INSTALL      VERSION.in  icons/       src/
INSTALL.FAST acconfig.h  install-sh*  tk/
Make.common.in aclocal.m4  lib/         vfs/
Makefile.in  config.h.in mc.spec      xv/
NEWS         configure*  mc.spec.in
README       configure.in mcfn_install.in
README.NT*   create_vcs* nt/
[08] $ _
```

Das Kommando in Zeile 05 gehört zur Trickkiste der bash ([History Funktion](#)). Hier wird das vorhergehende Kommando wiederholt und dabei die Buchstaben `tv` aus der Optionenreihe von `tar` durch `x` ersetzt.

Die erste Aktion nach dem Auspacken ist natürlich das Listing des neu erzeugten Verzeichnisses. Die Option `-F` sorgt für die Markierung der ausführbaren Dateien und der Verzeichnisse, das erleichtert die Orientierung.

In dem Listing sind einige Dateien mit der Endung `.in` zu sehen. Zusammen mit der Datei `configure` ist das ein deutlicher Hinweis darauf, daß dieses Paket mit GNU-autoconf  konfiguriert wird. Die `.in`-Dateien werden durch die Konfigurationsprozedur automatisch an das lokale System angepaßt.

Weil jedes Softwarepaket seine Eigenarten hat, ist es sinnvoll, zuerst einmal die wichtigsten Dokumente durchzusehen. Zum Lesen der Dateien eignen sich am besten Pager wie [more](#) oder `less`. Als erstes sollte immer die Datei `README` (auch `Readme`, `Read.Me`, `Liesmich` oder ähnlich) gelesen werden. Wenn eine frühere Version des Programms bereits bekannt ist, bieten sich die `NEWS` an. Als nächstes drängt sich die Datei `INSTALL` auf, oder, weil wir es eilig haben, besser noch `INSTALL.FAST`.


Außerdem zeigt das Listing von Zeile 07 ein Verzeichnis `doc/`, in dem sich vermutlich weitere Information zum Programm befindet.

```
[08] $ less README
[09] $ less INSTALL.FAST
...
1. Configure the package for your system.

    Normally, you just `cd' to the package main directory and type
`./configure'.
...
2. Type `make' to compile the package.

3. Type `make install' (as root) to install programs, data files, and
documentation.
...
[10] $ ls -F doc/
DEVEL          Makefile.in          mc.sgml
FILES          linuxdoc-sgml.diff   mcedit.1
LSM            mc.1                 mcserv.8
[11] $ _
```

Der kürzeste Weg zum ausführbaren Programm ist also: `./configure` eingeben, dann `make` und dann `make install`.

Das Listing des Dokumentationsverzeichnisses zeigt einige Manualpages,  die bei der späteren Installation des Programms in ein Verzeichnis des `MANPATH` kopiert werden. Die Datei `mc.sgml` enthält Dokumentation zum Midnight Commander mit Formatanweisungen in der *Standard Generic Markup Language*. Zum Formatieren dieses Dokuments wird ein Paket namens `linuxdoc-sgml` benötigt. Die Datei `linuxdoc-sgml.diff` enthält eine Verbesserung für dieses Paket, die zum Formatieren von `mc.sgml` benötigt wird. Mit dem Programm `patch` können die Änderungen an `linuxdoc-sgml` automatisch durchgeführt werden. All diese Dateien sind aber für die Konfiguration, das Übersetzen und den Test des Programms nicht notwendig. Für eine intensivere

Beschäftigung mit den Sourcen ist möglicherweise noch die Datei `FILES` interessant, in der eine kurze Inhaltsangabe aller zum Paket gehörenden Dateien zu finden ist.

Nachdem zunächst die wichtigsten Informationen gesichtet wurden, kann mit der eigentlichen Konfigurierung des Softwarepakets begonnen werden.

```
[10] $ ./configure
creating cache ./config.cache
checking for prefix by ... checking for mc... /usr/local/bin/mc
checking whether make sets ${MAKE}... yes
checking for gcc... gcc
checking whether we are using GNU C... yes
...
```

Configuration:

```
Source code location:      .
Compiler:                  gcc
Compiler flags:            -g -O
File system:               Midnight Commander Virtual File System
                           tarfs, mcfs, ftpfs
Text mode screen manager:  SLang with terminfo
Install console saver:     yes
Text mode mouse library:   GPM and xterm
Debugger code:             none
With subshell support:     yes
X11 versions:              none
Internal editor:           yes
Install path:              /usr/local/bin, /usr/local/lib/mc
```

```
[11] $ _
```

Diese Konfiguration ist zum Testen des neuen Programms sehr gut. Die Kombination `-g -O` für die Compiler Flags sorgt dafür, daß spezielle Informationen zum Debuggen in das Programm eingebaut werden (durch die Compileroption `-g`) und daß der Compiler das Programm einfach optimiert (durch die Compileroption `-O`). Als Installationspfad ist `/usr/local/bin` und `/usr/local/lib/mc` festgelegt. Diese Verzeichnisse befinden sich außerhalb des Bereiches, in dem die Linux-Distributionen ihre Programme installieren. Deshalb kann das neue Programm hier problemlos installiert werden, ohne daß die Gefahr besteht, ein bestehendes Softwarepaket zu stören.

Wäre das neue Programm fertig getestet und sollte es anstelle einer vielleicht bereits installierten früheren Version in `/usr/bin` installiert werden, könnte `configure` auch folgendermaßen aufgerufen werden:

```
[10] $ CFLAGS="-O2" ./configure --prefix=/usr
...
[11] $ _
```

Durch diese Konfigurationsvariante werden mit den `CFLAGS` die Compileroptionen so eingestellt, daß keine Debuginformationen mehr in die Binärdatei eingebaut werden. Stattdessen wird der Optimierungsgrad erhöht. Außerdem wird der Installationspfad durch die Angabe eines `-prefix` in den Stamm des Verzeichnisbaums verlegt.

Softwarepakete, die sich nicht mit `GNU-autoconf` konfigurieren lassen, müssen "von Hand" an das



System angepaßt werden. Wenn in den Dokumenten kein Hinweis auf die Konfiguration für Linux zu finden ist, reicht meistens die Anlehnung an ein anderes UNIX-Modell aus. In der Regel eignet sich System V (häufig auch SYSV, SysV, SVR3 oder ähnliches) am besten. Für die Anlehnung an den BSD-Stil muß eventuell die Kompatibilitätsbibliothek `libbsd.a` durch die Compileroption `-lbsd` hinzugelinkt werden.

Auch nach der Konfiguration mit `autoconf` ist es manchmal sinnvoll, kurz einen Blick in die Datei `Makefile` zu werfen. Möglicherweise lassen sich dort noch systemspezifische Anpassungen vornehmen.

Speziell nach der automatischen Konfiguration verschafft das Kommando `'ls -rtl'` nochmal einen guten Überblick. Das Listing wird, nach der Modifikationszeit sortiert, mit der jüngsten Datei zuletzt ausgegeben. So läßt sich leicht erkennen, welche Dateien von `configure` erzeugt oder verändert wurden.

```
[11] $ ls -rtl
...
drwxr-xr-x  4 she      she      1024 Sep 16 23:55 os2
-rw-rw-r--  1 she      she      5288 Oct  5 12:28 config.log
-rw-rw-r--  1 she      she      7290 Oct  5 12:28 config.cache
-rw-rw-r--  1 she      she      2850 Oct  5 12:28 Make.common
-rwxrwxr-x  1 she      she     23511 Oct  5 12:28 config.status
drwxr-xr-x  2 she      she      1024 Oct  5 12:28 doc
drwxr-xr-x  3 she      she      1024 Oct  5 12:28 vfs
-rw-rw-r--  1 she      she     3875 Oct  5 12:28 Makefile
drwxr-xr-x  2 she      she      1024 Oct  5 12:28 lib
drwxr-xr-x  3 she      she      1024 Oct  5 12:28 xv
drwxr-xr-x  2 she      she      1024 Oct  5 12:28 tk
drwxr-xr-x  2 she      she     2048 Oct  5 12:28 src
drwxr-xr-x  2 she      she      1024 Oct  5 12:28 slang
drwxr-xr-x  2 she      she      1024 Oct  5 12:28 edit
drwxr-xr-x  2 she      she     2048 Oct  5 12:28 icons
-rw-rw-r--  1 she      she     1877 Oct  5 12:28 mcfn_install
-rw-rw-r--  1 she      she     12733 Oct  5 12:28 config.h
[12] $ _
```

Die interessanteste Datei in diesem Listing ist wahrscheinlich `config.h`. Sollte es zu Problemen beim übersetzen des Programms kommen, ist das ein guter Ort, um mit der Suche nach der Ursache zu beginnen.

Nach all den Vorbereitungen ist es Zeit für einen ersten Versuch, das Programm zu Übersetzen.

```
[12] $ make
make[1]: Entering directory `/usr/src/mc-4.1/vfs'
DLIBDIR=\"/usr/local/lib/mc/\" -DICONDIR=\"/usr/local/lib/mc/icons/\" -D
HAVE_CONFIG_H -g -O local.c
gcc -c -I.. -I../vfs -I../slang -I.. -DBINDIR=\"/usr/local/bin/\"
\" -DLIBDIR=\"/usr/local/lib/mc/\" -DICONDIR=\"/usr/local/lib/mc/icons/\"
-DHAVE_CONFIG_H -g -O vfs.c
...
make[1]: Nothing to be done for `all'.
make[1]: Leaving directory `/usr/src/mc-4.1/xv'
make[1]: Entering directory `/usr/src/mc-4.1/icons'
make[1]: Nothing to be done for `all'.
make[1]: Leaving directory `/usr/src/mc-4.1/icons'
[13] $ _
```



Ganz unspektakulär wird der Compilerlauf nach wenigen Minuten erfolgreich abgeschlossen. Genauere Erläuterung zu den während der Übersetzung ausgegebenen Meldungen gibt es bei der Dokumentation zu `make` und `gcc`.

Ein Fehler bei der Übersetzung hätte sich deutlich zu erkennen gegeben:

```
...
main.c: In function `do_execute':
main.c:802: `last_passed' undeclared (first use this function)
main.c:802: (Each undeclared identifier is reported only once
main.c:802: for each function it appears in.)
make[1]: *** [main.o] Error 1
make[1]: Leaving directory `/usr/src/mc-4.1/src'
make: *** [all] Error 1
[13] $ _
```

Wie mit einem Fehler umzugehen ist, muß von Fall zu Fall unterschiedlich entschieden werden. Hier fängt das Abenteuer an. Debugging kann so spannend sein wie ein Adventure-Game. Das schönste ist aber, daß am Ende etwas dabei herauskommt, das von vielen tausend Usern auf der realen Seite unserer virtuellen Welt benutzt werden kann.

Auch ohne die Hilfe der großartigen Entwicklungswerkzeuge `grep`, `gdb`, `nm`, `strace` oder `syslogd` kann der kleine Fehler oben behoben werden: `last_passed` muß in `last_paused` geändert werden, dann klappt alles.

Nachdem die Übersetzung erfolgreich abgeschlossen ist, kann die erste Testinstallation durchgeführt werden. Eventuell kann das neue Programm auch ohne Installation direkt aus dem Sourceverzeichnis heraus aufgerufen werden. Im Fall des Midnight Commanders fehlen dann die Hilfsdateien, bestimmte Funktionen werden deshalb mit entsprechenden Fehlermeldungen abgeschlossen.

Um die Installation in `/usr/local` durchführen zu können, muß zunächst in eine Shell mit Superuser-Rechten gewechselt werden. Dann kann durch das Kommando `'make install'` der letzte Schritt ausgeführt werden.

```
[13] $ su
Password:
[14] # make install
./src/xmkdir /usr/local/bin /usr/local/lib/mc
mkdir /usr/local/lib/mc
./src/xmkdir /usr/local/man/man1 /usr/local/man/man8
...
make[1]: Leaving directory `/usr/src/mc-4.1/icons'
/usr/bin/install -c -m 644 ./FAQ /usr/local/lib/mc/FAQ
/usr/bin/install -c mcfnc_install /usr/local/lib/mc/bin/mcfnc_install
chmod +x /usr/local/lib/mc/bin/mcfnc_install
Please verify that the configuration values are correctly
set in the mc.ext file in /usr/local/lib/mc
[15] # exit
[16] $ _
```

Am Ende der Installation wird auf die Konfigurationsdatei `mc.ext` hingewiesen. Sie enthält sinnvolle Voreinstellungen, die eventuell an das lokale System angepaßt werden können/müssen.

Mit diesem letzten Schritt ist die Installation abgeschlossen. Der Weg vom Binärpaket zum

ausführbaren Programm ist erfolgreich zurückgelegt. Nun folgt eine Phase intensiven Testens des Programms. Im Betrieb taucht vielleicht der eine oder andere Fehler im Programm auf. Da die Sourcen ausgepackt sind, läßt sich eine korrigierte Version sehr schnell erzeugen.

Wenn sich das Programm als stabil erweist, kann das Programm im Stamm des Verzeichnisbaums installiert und das Sourceverzeichnis mit dem Kommando 'make clean' aufgeräumt werden.

## Paketmanagement mit RPM

Wie der vorangegangene Abschnitt zeigt, ist die Herstellung eines Binärprogramms aus einem Sourcepaket keine unüberwindliche Hürde, wenn es darum geht, eine wesentlich verbesserte Programmversion oder gar ein ganz neues Programm zu installieren.

Es dauert aber in der Regel nur ein paar Tage, bis irgendwo im Internet aus dem Sourcepaket einer neuen Softwareversion auch die passenden Binärpakete für die verschiedenen Linux-Distributionen entstanden sind. Um ein solches Binärpaket zu bekommen, müssen Sie nicht auf die nächste Distributions-CD warten. Alle Distributoren bieten auch Updates für ihre Produkte per FTP an. Die Installation oder das Upgrade eines solchen Binärpakets ist wesentlich bequemer als die Erzeugung der gleichen Software aus den Sourcen.

## Installation und Upgrade einer Software

Mit dem RPM-System reduziert sich der Aufwand für das Upgrade eines RPM-Pakets auf einen einzigen Aufruf des Programms [rpm](#):

```
[01] # rpm -q mc
mc-3.2.1-1
[02] # rpm -U ftp.tu-clausthal.de/pub/linux/redhat/i386/mc-4.1.3-1.i386.rpm
[03] # echo $?
0
[04] # rpm -q mc
mc-4.1.3-1
[05] #
```

Das entscheidende Kommando steht in Zeile 02. Der Rest dient allein dem Nachweis, daß das Paket tatsächlich erneuert wurde.

Durch Verwendung der Option `-i` anstelle von `-U` würde `rpm` im Installationsmodus arbeiten.

Ein Upgradeversuch mit einem RPM-Binärpaket kann aber auch leicht fehlschlagen. Der RedHat Packet Manager `rpm` prüft vor Installation oder Upgrade, ob alle für das einwandfreie Funktionieren der Software erforderlichen Services anderer Pakete bereitgestellt werden. Solche Services sind beispielsweise die Shared Libraries für dynamisch gelinkte Programme. Es können aber auch andere Programme sein, die von der neuen Software im Hintergrund aufgerufen werden.

Beim Upgrade auf neue Softwareversionen kann zusätzlich das Problem auftreten, daß ein spezieller Service der alten Version von anderen Paketen benötigt wird, diese Version also nicht gelöscht

werden darf. Ein Beispiel für diese Situation wird weiter unten gegeben.

Zu den häufigsten Problemen bei Installation oder Upgrade gehört das Fehlen der neuesten Version einer Shared Library. Das sieht dann so aus:

```
[01] # rpm -U mc-4.1.3-1.i386.rpm
failed dependencies:
    libslang.so.0 is needed by mc-4.1.3-1
    libncurses.so.3.0 is needed by mc-4.1.3-1
[02] # rpm -q mc
mc-3.2.1-1
[03] # _
```

So ein Fehler muß nicht unbedingt bedeuten, daß die Software nicht installiert werden kann. Bei den Distributionen von LST und Caldera wird jedenfalls bis zur Version 1.1 die gegenseitige Abhängigkeit der Pakete nicht berücksichtigt. Das hat zur Konsequenz, daß in der Datenbank, in der rpm normalerweise die Services der im System installierten Pakete verzeichnet, kein einziger Service eingetragen ist. Das bedeutet nicht, daß die Services nicht vorhanden sind. Es hat aber zur Folge, daß jedes Paket, das irgendeinen Service anfordert, von rpm abgewiesen wird.

In einem solchen Fall kann die Installation durch die Option `-nodeps` erzwungen werden:

```
[03] # ls /lib/libncurses.so.3.0 /usr/lib/libslang.so.0
/lib/libncurses.so.3.0  /usr/lib/libslang.so.0
[04] # rpm -U --nodeps mc-4.1.3-1.i386.rpm
[05] # rpm -q mc
mc-4.1.3-1
[06] # mc
[07] # _
```

## Untersuchung von RPM-Paketen

Mit dem Query-Modus von rpm können die RPM-Pakete auch vor dem Installationsversuch untersucht werden. Die Probleme mit der Abhängigkeit von anderen Paketen können dann ebenfalls systematisch gelöst werden:

```
[01] # rpm -q --requires -p mc-4.1.3-1.i386.rpm
/usr/bin/perl
/bin/sh
libslang.so.0
libncurses.so.3.0
libm.so.5
libgpm.so.1
libc.so.5
[02] # rpm -q --whatprovides libc.so.5
libc-5.3.12-18
[03] # rpm -q --whatprovides libncurses.so.3.0
no package provides libncurses.so.3.0
[04] # rpm -q --provides -p ncurses-4.1-3.i386.rpm
libpanel.so.4
libncurses.so.4
libmenu.so.4
libform.so.4
[06] # rpm -U ncurses-4.1-3.i386.rpm
failed dependencies:
    libncurses.so.3.0 is needed by mc-4.1.3-1
```

```
[07] # _
```

Die Zeile 06 zeigt, wie rpm verhindert, daß eine von anderen Paketen benötigte Datei beim Upgrade eines RPM-Pakets verloren geht. Im Fall von Shared Libraries wie bei ncurses ist das nicht unbedingt die beste Lösung. Schließlich können unter Linux durchaus mehrere verschiedene Major-Versionen einer Shared Library nebeneinander existieren. Insgesamt überwiegen die Vorteile des RPM-Systems aber solche Problemfälle bei weitem.

## Direkter Zugriff auf die Daten in der RPM-Datei

Um nach dem erzwungenen Upgrade auf die neue Version eine einzelne Datei aus dem Archiv des alten Pakets zu extrahieren, kann das Programm rpm2cpio zusammen mit [cpio](#) verwendet werden.

```
[08] # rpm -q -lv -p ncurses-1.9.9g-7.i386.rpm | grep libncurses
-rwxr-xr-x-      root      root      262468 Nov 26 00:33 /lib/libncurses.so.1.9.
9g
lrwxrwxrwx-      root      root              20 Nov 26 00:34 /lib/libncurses.so.2.0
-> libncurses.so.1.9.9g
lrwxrwxrwx-      root      root              20 Nov 26 00:34 /lib/libncurses.so.3.0
-> libncurses.so.1.9.9g
lrwxrwxrwx-      root      root              20 Nov 26 00:33 /lib/libncurses.so.3.2
-> libncurses.so.1.9.9g
[09] # rpm2cpio ncurses-1.9.9g-7.i386.rpm | cpio -id ``lib/libncurses.so.*''
[10] # cp lib/* /lib
[11] # ldconfig
[12] # _
```

Die Scriptdateien für Vorbereitung und Abschluß der Installation bzw. Löschung werden im Query-Modus durch die Option `-scripts` ausgegeben.

## RPM und die Konfiguration der Software

Bei vielen Softwarepaketen ist über die eigentliche Installation hinaus noch eine systemspezifische Anpassung notwendig. Dazu werden in der Regel Einträge in Konfigurationsdateien erzeugt oder verändert.

Vor und nach dem Auspacken der Paketdaten führt rpm Shellscripts aus, die gewisse Konfigurationsarbeiten automatisch durchführen können. Andere Systemeinstellungen müssen von der Systemverwalterin ``manuell" vorgenommen werden.

Voraussetzung für die Durchführung dieser Arbeit ist natürlich die Kenntnis, welches die Konfigurationsdateien sind und wo sie sich befinden. rpm kann diese Identifikation sehr leicht vornehmen. Außerdem kann es die zu einer Software gehörende Dokumentation ausfindig machen. Für beide Dienste muß rpm im Query-Modus arbeiten:

```
[01] # rpm -q -c sendmail
/etc/aliases
/etc/aliases.db
/etc/rc.d/init.d/mta
/etc/sendmail.cf
/etc/sendmail.cw
```

```
/etc/sysconfig/daemons/mta
[02] # rpm -q -d sendmail
/usr/man/man1/mailq.1.gz
/usr/man/man1/newaliases.1.gz
/usr/man/man5/aliases.5.gz
/usr/man/man8/makemap.8.gz
/usr/man/man8/rmail.8.gz
/usr/man/man8/sendmail.8.gz
[03] # _
```

Diese Hilfestellung ist zwar nur ein kleiner, möglicherweise aber wesentlicher Schritt zur erfolgreichen Integration einer Software in ein bestehendes System.

Ist ein Softwarepaket einmal installiert und konfiguriert, bewahrt rpm die systemspezifischen Anpassungen auch beim Upgrade auf eine neuer Version der Software, sogar beim Löschen eines Pakets läßt das Programm alle veränderten Konfigurationsdateien zurück. Die gesicherten Dateien haben die Endung `.rpmsave` und liegen im gleichen Verzeichnis, in dem auch die Konfigurationsdatei gelegen hat.

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [Datensicherung](#) **Up:** [Systemverwaltung](#) **Previous:** [Partitionen und Dateisysteme](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

## Subsections

- [Medien zur Datensicherung](#)
    - [Magnetbänder im Allgemeinen](#)
    - [Mehrere Dateien \(Archive\) auf einem Magnetband](#)
    - [Dateien \(Archive\) auf mehreren Magnetbändern](#)
    - [Floppystreamer](#)
    - [QIC-Streamer](#)
    - [SCSI-Streamer](#)
    - [Disketten](#)
  - [Methoden der Datensicherung](#)
  - [Backup Software](#)
    - [Datensicherung mit tar](#)
    - [Datensicherung mit afio und cpio](#)
- 

# Datensicherung

Die regelmäßige Datensicherung (Backup) ist eine der wichtigsten Aufgaben der Systemverwalterin. Sei es durch versehentliches Löschen, sei es durch Hardwarefehler oder durch einen Fehler des Betriebssystems, früher oder später macht jeder Computerbenutzer die Erfahrung eines Datenverlustes. Ein Backup ist in der Regel die einzige Möglichkeit, wenigstens einen Teil der Daten zu restaurieren.

Auch wenn ein regelmäßiges Backup einigen Arbeitsaufwand bedeutet, steht diese Mühe meist in keinem Verhältnis zu einer manuellen Rekonstruktion der Daten. Auf professionellen Systemen, bei denen möglicherweise sogar mehrere Benutzer auf einem Datenbestand arbeiten, ist wenigstens eine wöchentliche, besser eine tägliche Datensicherung erforderlich.

## Medien zur Datensicherung


Datensicherung kann auf sehr verschiedene Weisen und auf verschiedenen Medien erfolgen. Es besteht zum Beispiel die Möglichkeit, Kopien von wichtigen Dateien in einem anderen Teil des Dateisystems (auf einer anderen Partition oder Festplatte) anzulegen. Natürlich können auch Disketten zur Datensicherung verwendet werden. Im allgemeinen werden aber Magnetbänder benutzt, um Backups vom System zu machen.

## Magnetbänder im Allgemeinen

Magnetbänder haben gegenüber allen anderen Medien mehrere Vorteile:

- Sie bieten eine hohe Kapazität zu einem niedrigen Preis.
- Sie lassen sich leicht vom Rechner entfernen und an einem sicheren Ort aufbewahren.
- Sie eignen sich gut zur unbeaufsichtigten Datensicherung.

Der Nachteil von Magnetbändern besteht im sehr langsamen Zugriff auf einzelne Dateien. Die Topologie des Magnetbandes erzwingt eine rein sequentielle Form der Datenspeicherung. Um an Daten in der Mitte des Bandes heranzukommen, muß das gesamte Band bis zur gesuchten Stelle gelesen werden. Die Vorteile überwiegen diesen Nachteil bei der Datensicherung aber so stark, daß auf allen Systemen, wo ernsthafte Datensicherung betrieben wird, Magnetbänder eingesetzt werden.

Magnetbandlaufwerke werden als zeichenorientierte Geräte betrieben. Trotzdem sind die Daten auf dem Medium in Blöcken organisiert. Das schafft einige Verwirrung und führt in manchen Grenzfällen zu Komplikationen. Von den echten Blockgeräten unterscheiden sich die Bandlaufwerke, weil nicht ein bestimmter einzelner Block gelesen oder geschrieben werden kann.  Im Unterschied zu den Magnetplattenspeichern werden die Magnetbänder in der Regel nicht formatiert. Deshalb gibt es auf Magnetbändern keine physikalisch nummerierten Datenblöcke, es gibt keine Dateisysteme und kein Inhaltsverzeichnis und das Einbinden eines Magnetbandes in das Dateisystem mit dem mount-Kommando ist unmöglich. Der Zugriff auf die Daten findet immer sequentiell statt. Um die Daten eines bestimmten Blockes zu erhalten, müssen alle vorhergehenden Blöcke gelesen werden.

Bei den alten 9-Spur Industrielaufwerken konnte (oder mußte) das Band im Start/Stop Betrieb zwischen zwei Datenblöcken immer angehalten werden. Bei den Magnetbandgeräten für PC geht das in der Regel nicht, weil der für das Anhalten nötige Zwischenraum zugunsten der Datenkapazität praktisch weggefallen ist. Die Daten werden auf den Magnetbändern als kontinuierlicher Datenstrom gespeichert. Wegen dieser Art des Datentransfers werden die Bandlaufwerke auch als Streamer bezeichnet.

Für alle Magnetbandgeräte existieren zwei verschiedene Betriebsarten:

1.

``Rewind on Close''. In dieser Betriebsart wird das Band nach dem Schließen der Gerätedatei, also nach Beendigung der Lese- oder Schreiboperation, automatisch zurückgespult.

2.

``No Rewind on Close''. In dieser Betriebsart wird das Band nach Beendigung einer Operation angehalten und bleibt so stehen, bis die nächste Operation durchgeführt wird.

Die Auswahl einer Betriebsart findet durch die Gerätedatei statt, über die das Laufwerk angesprochen wird.

Viele Bandgeräte können mit Magnetbändern unterschiedlicher Kapazität arbeiten. Dazu müssen verschiedene Aufzeichnungsparameter, namentlich die Anzahl der Spuren, die Schreibgeschwindigkeit und die Schreibdichte, eingestellt werden.

Obwohl die meisten Laufwerke diese Parameter automatisch einstellen, erlauben einige Gerätetreiber die Vorauswahl eines Bandtyps durch die Benutzung einer bestimmten Gerätedatei.

Alle von Linux unterstützten Bandlaufwerke arbeiten mit Magnetbandkassetten (Cartridges). Die Bänder der am weitesten verbreiteten Streamer sind 1/4 Zoll breit, deshalb werden sie auch als Quarter-Inch-Cartridges bezeichnet. Sowohl das Aufzeichnungsformat für solche Bänder, als auch die Ansteuerung der mit diesen Bändern arbeitenden Laufwerke ist in den verschiedenen Standards des Quarter Inch Comitee (QIC) definiert.

Die für Linux wichtigsten QIC-Normen sind:

## QIC-02

In der QIC-02 Spezifikation ist die Ansteuerung von DC6XXX Streamern mit eigenen Controllerkarten beschrieben.

## QIC-24

In QIC-24 ist das Aufzeichnungsformat für DC600A Bänder mit 60MB Kapazität (9Spuren, 10.000FTPI) definiert.

## QIC-40/-80/-3010/-3020

In den Standards QIC-40, QIC-80 QIC-3010 und QIC-3020 sind die Aufzeichnungsformate für DC2XXX und Travan TR2/TR3 Magnetbänder festgelegt, wie sie in Floppystreamern verwendet werden.

## QIC-117

In QIC-117 ist die Ansteuerung der Floppystreamer beschrieben.


## QIC-120/-150/-525...

Die Spezifikationen QIC-120, QIC-150, QIC-525 usw. beschreiben das Aufzeichnungsformat der DC6XXX Magnetbänder mit den entsprechenden Kapazitäten. Das Aufzeichnungsverfahren ist bei diesen Formaten identisch mit QIC-24, lediglich die Aufzeichnungsdichte und die Bandqualität sind unterschiedlich.

Neben den QIC-konformen Bandgeräten kann Linux auch mit allen SCSI-Streamern  umgehen.

Alle Streamer, die am Druckerport betrieben werden, können unter Linux (noch) nicht betrieben werden.

## Mehrere Dateien (Archive) auf einem Magnetband

Wenn ein kontinuierlicher Datenstrom, sprich eine Datei, abgeschlossen ist, wird auf dem Band eine Markierung für das Dateiende (EOF, End Of File)  geschrieben. Wenn Sie ein Bandarchiv mit `tar` oder `cpio` erzeugen, wird diese Markierung immer ans Ende des gesamten Archivs gesetzt. Die einzelnen Dateien in diesem Archiv werden nur durch das Archivierungsprogramm unterschieden.

Wenn nach dem Dateiende noch freier Speicherplatz auf dem Band vorhanden ist, kann eine weitere Datei oder ein Archiv an die bereits geschriebenen Daten angehängt werden.

Mit dem `mt`-Kommando müssen Sie dazu das Band hinter die Endmarkierung positionieren. Damit das Band nach dem `mt`-Kommando nicht automatisch zurückgespult wird, muß das Bandgerät im ``No Rewind on Close" Modus betrieben werden.

Wenn Sie beispielsweise bereits eine Datei (ein Archiv) auf einem SCSI-Streamerband gespeichert haben und eine zweite Datei auf das gleiche Band schreiben wollen, positionieren Sie das Band mit dem folgenden Kommando hinter die erste Datei:

```
$ mt -f /dev/nst0 fsf 1
$ _
```

Mit der `-f` Option wird der erste SCSI-Streamer im ``No Rewind On Close" Modus ausgewählt. Die Operation `fsf 1` (forward skip file) spult das Band bis zur ersten Dateiendemarke vor. Wenn das Kommando abgeschlossen ist, hält der Bandmotor an und der Schreib/Lesekopf steht hinter der ersten




Datei. Wenn Sie jetzt mit einem der Archivierungsprogramme ein Schreib- oder Lesekommando ausführen, läuft der Bandmotor wieder an und führt die Aktion an dieser Stelle aus.

Wenn mehrere Dateien (Archive) auf einem Band gespeichert sind, können Sie keine Datei am Anfang oder in der Mitte löschen/überschreiben. Eine Schreiboperation in der Mitte des Bandes würde automatisch alle hinter der Dateiende-Markierung liegenden Daten unzugänglich machen.


## Dateien (Archive) auf mehreren Magnetbändern

Obwohl die Magnetbänder große Datenmengen speichern können, kommt es manchmal vor, daß ein Linux-System oder eine Linux-Partition, nicht einmal komprimiert, auf ein einzelnes Magnetband paßt. In diesem Fall kann, die geeignete Software vorausgesetzt, das Backup auch auf mehrere Bänder verteilt werden.

Durch spezielle Markierungen auf den Magnetbändern, die sogenannten Early Warning Marks, erkennt das Bandgerät das Bandende im Voraus. Bei den Bandgeräten mit fester Blockgröße (QIC-02 und Floppystreamer) reicht die Kapazität immer aus, um den aktuellen Datenblock und einen abschließenden Block vom Archivierungsprogramm unterzubringen. Bei SCSI-Streamern, die variable Blockgrößen verwenden können und die einen Teil der Daten verzögert schreiben, kann es zu Problemen kommen, wenn das Band die verzögerten Daten nicht mehr fassen kann. 

## Floppystreamer

Wegen des günstigen Anschaffungspreises sind die Floppystreamer für den Einsatz in kleinen bis mittleren Linux-Systemen besonders interessant. Diese Bandlaufwerke werden wie ein drittes Diskettenlaufwerk an den normalen Floppycontroller angeschlossen.

Die im Handel befindlichen kleinen Floppystreamer verwenden in der Regel das QIC-80 Format. In diesen Streamern werden Mini-Cartridges vom Format DC2080 oder DC2120 verwendet, die eine Kapazität von 80 bzw. 120 Megabyte unkomprimierter Daten haben, durch extra lange Bänder kann die Kapazität auf 170 MB gesteigert werden.  Die moderneren Floppystreamer mit höheren Kapazitäten arbeiten mit Traven TR2/TR3-Bändern in den Formaten QIC-3010 und QIC-3020. Diese Bänder können 400MB und mehr speichern.

Die Datentransferrate aller Floppystreamer wird vom Floppycontroller bestimmt. Im schlechtesten Fall lassen sich 126 Megabyte pro Stunde schreiben. Mit dem Controllerbaustein 82078-1 kann die Leistung des gleichen Laufwerks vervierfacht werden.

Im Unterschied zu den SCSI-Streamern, die durch die sehr genaue Spezifikation der Geräteschnittstelle alle in der gleichen Weise angesteuert werden können, lassen sich die QIC-117 Streamer nicht alle exakt gleich betreiben. Für Linux gibt es Treiber für die folgenden Geräte:

---

Colorado DJ-10  
Colorado Jumbo 250  
Archive 5580i  
Insight 80Mb

Colorado Jumbo 120  
Summit SE 150  
Archive XL9250i  
Conner C250MQ


Colorado DJ-20  
Summit SE 250  
Archive 31250Q  
Wangtek 3080F

Zur Beschleunigung der Floppystreamer werden spezielle Adapterkarten mit einem schnellen Controllerbaustein angeboten. Ftape unterstützt folgende Adapter:

- Colorado FC-10, FC-20
- Mountain Mach-2
- Iomega Tape Accelerator II
- Iomega Ditto Dash
- i82078-1 basierte Floppycontroller

Seit der Treiber für Floppytapes in den normalen Kernelsourcen enthalten ist, kann beim Übersetzen eines neuen Kernels zwischen der festen Einbindung des Treibers in den Kernel oder der Übersetzung des Treibers als Modul gewählt werden.

Das Laden des Moduls kann auch durch den ``Kerneldämon" `kernel.d` bei Bedarf automatisch vorgenommen werden. In diesem Fall müssen Sie darauf Achten, das Gerät immer mit Rewind On Close zu benutzen. Anderenfalls kann beim automatischen Entfernen und Neuladen des Treibermodul ein Fehler in der Blockzählung auftreten und dadurch Daten verloren gehen.

Um Daten auf einem Floppytape speichern zu können, muß das Medium zuerst formatiert werden. Ähnlich wie beim Formatieren von Disketten werden leere Blöcke mit einer Größe von 512 Bytes auf das leere Band geschrieben. Diese Blöcke werden dann beim Beschreiben des Bandes mit Daten gefüllt. 

Linux erlaubt bislang nicht die Formatierung von Magnetbändern. Wenn Sie Ihre Floppytapes nicht unter MS-DOS formatieren können oder wollen, sollten Sie sich einfach fertig formatierte Markenbänder kaufen. Das spart erstens viel Zeit und stellt zweitens eine hohe Qualität des Magnetbandes sicher.

## QIC-Streamer

Die Bandlaufwerke mit einem eigenen Controller nach dem QIC-02 Standard verwenden 1/4 Zoll Cartridges normaler Größe (15x10 cm). Je nach Streamertyp können die QIC-02 Geräte Bänder in den Formaten QIC-24, QIC-120, QIC-150 usw. schreiben und/oder lesen.

Der QIC-02 Treiber wurde für den Wangtek-5150 Streamer geschrieben. Es gibt keine offizielle Liste aller unterstützten QIC-02 Streamer. Bei Everex und Archive Laufwerken sind die Chancen sehr gut, daß der Streamer sofort erkannt wird.

Je nach Typ des Bandgerätes können Bänder mit verschiedener Aufzeichnungsdichte verwendet werden. Die Einstellung einer bestimmten Dichte geschieht durch die Wahl der Gerätedatei für das Bandgerät.

Die QIC-02 Streamer arbeiten mit einer festen Blockgröße von 512 Bytes.

## SCSI-Streamer

Wie alle SCSI Geräte haben auch die SCSI Bandlaufwerke eine sehr umfangreiche Steuerlogik ``an

Bord". Das Betriebssystem kommuniziert mit dem Laufwerk über den Hostadapter auf einem sehr hohen Abstraktionsniveau. Das Magnetbandtyp oder das physikalische Aufzeichnungsformat spielen hier keine Rolle mehr. Aus diesem Grund können praktisch alle SCSI Bandgeräte unter Linux eingesetzt werden. Lediglich die Größe der physikalischen Blöcke darf die maximale Puffergröße von 32kB nicht überschreiten.

Weite Verbreitung haben die 1/4 Zoll Streamer, die mit den gleichen Cartridges arbeiten wie die QIC-02 Bandgeräte und die DAT-Streamer, die auf nur 4mm breiten Bändern enorme Datenmengen speichern können.

Weil der SCSI-Standard sehr generell gefaßt ist, werden die physikalischen Aufzeichnungsparameter nicht fest vorgegeben.

Die Aufzeichnungsdichte, die Blockgröße und die Datenpufferung können durch Systemaufrufe verändert werden, vorausgesetzt die Kombination von Bandgerät und Band erlauben die gewünschten Werte.

Während die Aufzeichnungsdichte automatisch erkannt wird und in der Regel nicht verändert werden sollte, kann es bei manchen Bandgeräten nötig oder vorteilhaft sein, die Datenpufferung einzuschalten, um ein gleichmäßigeres Strömen der Daten zu erreichen.

Wenn Sie Magnetbänder zwischen verschiedenen Betriebssystemen tauschen wollen, muß die Einstellung für die Größe der physikalischen Datenblöcke beim schreibenden und dem lesenden System übereinstimmen. Bei den SCSI-Geräten können verschiedene feste oder während der Aufzeichnung variierende Blockgrößen benutzt werden, wenn das Bandformat und das Gerät mitspielen.

Die Datenübertragungsrate der SCSI-Streamer hängt stark vom verwendeten Bandgerät, der Aufzeichnungsdichte (also dem Bandtyp) und auch vom Hostadapter ab. Sie ist aber in jedem Fall deutlich höher als bei den Floppystreamern.

## **Disketten**

Anstelle von Magnetbändern können auch Disketten als Medium zur Datensicherung eingesetzt werden. Dabei werden die formatierten Disketten direkt, roh beschrieben. Ein Dateisystem ist ebenso unnötig wie das Mounten der Diskette. Stattdessen wird die rohe Diskette direkt über die Gerätedatei für das Diskettenlaufwerk angesprochen.

Im Unterschied zu den zeichenorientierten Bandlaufwerken arbeiten die Diskettenlaufwerke blockorientiert. Dadurch sind Rückschritte und Positionierung vor das Dateiende kein Problem.

## **Methoden der Datensicherung**

Das Ziel jeder Datensicherung ist es, alle System- und Benutzerdaten einer Linux-Installation vor einem ungewollten Verlust zu schützen.

Weil sich bestimmte Daten, vor allem die Daten der Systembenutzer, kontinuierlich ändern, ist eine hohe Frequenz der Datensicherung erforderlich. Für Systeme im professionellen Einsatz ist ein tägliches Backup angebracht.

Bei einer sehr hohen Backupfrequenz ist es weder erforderlich, noch wünschenswert, jedesmal

sämtliche Daten erneut zu sichern. Im Prinzip reicht es aus, immer nur die seit dem letzten Backup veränderten Dateien zu sichern, um den aktuellen Zustand restaurieren zu können. Diese Methode des inkrementellen (aufsteigenden) Backups hat mehrere Vorteile:

1.

Jedes einzelne Backup läßt sich sehr schnell durchführen, weil die Menge der veränderten Daten verglichen mit dem gesamten System sehr klein ist.

2.

Die Methode ist kostengünstig, weil keine unveränderten Daten doppelt gespeichert werden.

3.

Die Wahl des Mediums ist maximal flexibel. Bei vielen Systemen kann für die inkrementellen Backupsschritte sogar eine Diskette verwendet werden.

Bei inkrementellen Backups können noch verschiedene Level unterschieden werden, indem mehrere Sicherungen als Bezugspunkt gewählt werden. Einem vollständigen Backup wird der Level 0 zugeordnet. In den Backups mit Levels größer als 0 werden nur die Daten gespeichert, die seit der letzten Sicherung mit einem Level kleiner oder gleich dem aktuellen Level verändert worden sind. In einem Backup Level 1 werden also alle Daten gespeichert, die seit dem letzten Vollbackup verändert worden sind und die noch nicht in einem anderen Backup vom Level 1 enthalten sind. Im Level 2 werden dann nur die Daten gespeichert, die seit dem letzten Backup mit Level 0, 1 oder 2 verändert worden sind und so weiter.

Als Abwandlung des oben beschriebenen Modells können die Level auch so definiert werden, daß immer nur die Veränderungen seit einem Backup mit streng niedrigerem Level gespeichert werden. Bei dieser Methode werden bei wiederholten Sicherungen eines Levels viele Daten mehrfach gesichert.

Sie sollten in jedem Fall so viele Bänder zur Datensicherung verwenden, daß Sie bei jedem Sicherungsschritt das letzte Backup vollständig behalten können. Nur so sind Sie in der Lage, einen Festplattencrash während des Sicherungslaufes zu restaurieren.

Die optimale Methode der Datensicherung hängt von der Beschaffenheit Ihres Systems ab. In der Regel besteht sie aus zwei oder dreistufigen Kombinationen von vollständigen und inkrementellen Backups. In regelmäßigen, größeren Abständen wird eine vollständige Sicherung aller Daten auf einem oder mehreren Bändern großer Kapazität gemacht. Relativ zu dieser Vollsicherung werden dann inkrementelle Sicherungen durchgeführt.

---

Größe (MB)	Mountpunkt	
20	/	Rootpartition
120	/usr	Programme und Daten (statisch)
40	/home	Heimatverzeichnisse
60	/var/spool	News und Mail

---

Wenn Sie zum Beispiel eine Systemaufteilung wie in der Tabelle dargestellt mit einem QIC-80 Floppystreamer sichern wollen, bietet es sich für ein vollständiges Backup (Level 0) an, die /usr Partition auf einem Band und den Rest des Systems auf einem anderen zu sichern.

Wenn Sie dieses Vollbackup an jedem ersten Sonntag im Monat durchführen, können Sie auf einem dritten Band jeden weiteren Sonntag alle Änderungen der vergangenen Woche als inkrementelle

Sicherungen (Level 1) speichern. Um die Bandkapazität besser auszunutzen, können Sie alle Level 1 Archive eines Monats hintereinander auf ein Band schreiben.

Ein anderes Modell ergibt sich, wenn Sie die inkrementellen Sicherungen immer relativ zum letzten Vollbackup anlegen. Dann werden die Level 1 Sicherungen von Mal zu Mal größer. Sie machen das nächste Vollbackup, wenn die Menge der veränderten Daten ein von Ihnen bestimmtes Maß überschreitet, spätestens wenn sie nicht mehr auf einem einzigen Band Platz findet. Der Vorteil dieses Modells besteht darin, daß Sie das Magnetband nicht hinter die bereits geschriebenen Daten der letzten Sicherungen positionieren müssen, weil die ja noch einmal geschrieben werden. Der Nachteil ist, daß die Sicherungen von Mal zu Mal länger dauern.

Wenn Sie besonders wichtige Daten, beispielsweise die Heimatverzeichnisse, häufiger sichern wollen, bieten sich auf einem kleinen System Disketten als Sicherungsmedium an.

Weil Sie den größten Teil des Systems auf einem Installationsmedium vorliegen haben, ist im Extremfall gar keine Sicherung des kompletten Dateisystems notwendig. Sie können der ``rohen'' Linux-Installation den Level 0 zuordnen und alle Veränderungen als inkrementelle Sicherungen relativ zu Ihrer Distribution speichern.

Selbstverständlich stehen Ihnen Kommandos zur Verfügung, die Sie bei der Erstellung inkrementeller Backups unterstützen. Vor allem das [find-Kommando](#) ist hervorragend zur Erstellung von Dateilisten geeignet. Das tar-Programm bietet Ihnen Optionen, mit denen Sie sehr einfach inkrementelle Backups machen können. Es ist leicht möglich, die Datensicherung automatisch durch ein Script ausführen zu lassen.

## Backup Software

Auf Magnetbändern können keine Dateisysteme eingerichtet werden, sie enthalten keine Verzeichnisse und können nicht in den Dateisystembaum eingebunden werden. Es ist zwar möglich, mehrere Dateien auf ein Band zu schreiben, die Dateinamen, Eigentümer, Zugriffsrechte und alle Zeitmarken gehen aber verloren, weil diese Daten nicht Teil der Datei selbst sind, sondern in der Inode der Datei gespeichert werden.

Deshalb werden zur Sicherung der Daten auf Band spezielle Programme verwendet, die eine Vielzahl von Dateien zu einem einzigen Datenstrom, also einer einzigen Datei, zusammenfassen und mit allen dazugehörenden Systemdaten verwalten können.

Die am häufigsten zu diesem Zweck herangezogenen Programme sind tar, afio und cpio. Die Programme dump und restore, die bei BSD-Unix zur Datensicherung verwendet werden, sind unter Linux noch nicht erhältlich. In besonderen Fällen kommt noch das dd-Kommando als Sicherungssoftware in Frage.

## Datensicherung mit tar

Eine der Grundaufgaben von Backupsoftware, nämlich die Zusammenfassung mehrerer Dateien zu einer einzigen, sowie alle zum Management dieses Datenpaketes gehörenden Aufgaben werden nicht nur zur Datensicherung auf Magnetbändern benötigt. Die gleiche Funktionalität wird auch zur Verwaltung der C-Funktionsbibliotheken gebraucht. Diese Bibliotheken, auch als Archive bezeichnet, werden von dem zum Entwicklungssystem gehörenden ar-Kommando erzeugt und verwaltet.

Speziell auf die Verwendung mit Bandarchiven weiterentwickelt gibt es auf praktisch allen Unix-Systemen den Tape-Archiver tar. Dank des sehr flexiblen Dateikonzeptes von Linux kann tar

auch Archive im Dateisystem als neue Dateien erzeugen und sie verwalten. In dieser Form wird fast alle Freie Software im Internet verteilt. Die Softwarepakete aller Linux-Distributionen sind mit `tar` archiviert.


Eine komplette Kommandobeschreibung zu `tar` finden Sie [hier](#).

Mit den Magnetbandgeräten, die unter Linux Verwendung finden, können nicht alle Funktionen von `tar` uneingeschränkt genutzt werden. Insbesondere kann ein Bandarchiv nicht erweitert werden. Das bedeutet, daß die Optionen `-A`, `-r` und `-u` nicht funktionieren. Außerdem ist das Löschen oder Ersetzen einzelner Dateien in einem Bandarchiv mit `-delete` nicht möglich. Trotzdem ist das GNU-`tar` ausgezeichnet zur Sicherung/Archivierung großer Datenmengen auf Magnetbänder geeignet.


Sie erzeugen ein neues `tar`-Archiv mit der Option `-c`. Um beispielsweise ein vollständiges Backup auf das SCSI-Bandlaufwerk zu schreiben, geben Sie die folgende Kommandozeile ein:

```
# tar -c -M -b 126 -V "Level 0" -f /dev/rmt0 -g /var/adm/Backup/Dir /
tar: Removing leading / from absolute path names in the archive.
tar: Removing leading / from absolute links
Prepare volume #2 for /dev/rmt0 and hit return:
# _
```

Wie Sie sehen, macht `tar` automatisch aus allen absoluten Pfadnamen relative. Das geschieht, damit Sie das Archiv relativ zu jedem beliebigen Verzeichnis auspacken können. Wenn Sie die absoluten Namen behalten wollen, müssen Sie den Schalter `-P` setzen.

Die `-M` Option zeigt `tar` an, daß Sie ein Archiv auf mehreren Bändern anlegen wollen, weil Ihre Daten nicht auf ein einziges Band passen. Wenn Ihre Daten nicht mehr als zwei Bänder füllen, können Sie durch die `-z` Option die Daten durch den `gzip`-Kompressor filtern.  Wenn nicht ein großer Teil der Daten bereits komprimiert im Dateisystem vorliegt, erreichen Sie eine Verdichtung der Daten mindestens um den Faktor 2. Es ist allerdings nicht möglich, komprimierte `tar` Archive auf mehrere Bänder zu verteilen.

In dem Beispiel oben wurde die Blockgröße mit der Option `-b 126` gegenüber dem Standardwert von 20 deutlich erhöht. Das Optionsargument 126 bedeutet, daß `tar` die Daten in Portionen zu 126x512

Bytes, also 64 Kilobyte, auf das Band schreibt.  Die Vergrößerung des Wertes sorgt für einen flüssigeren Datenstrom und damit zu einer Beschleunigung des Sicherungslaufes. Sie müssen bei jeder Veränderung der Blockgröße aber darauf achten, daß Sie beim Lesen den gleichen Wert einstellen.

Mit der Option `-V` wird dem Archiv ein Name gegeben. Damit können Sie die Sicherungsbänder identifizieren, auch wenn Ihnen die externe Beschriftung verloren gehen sollte.

Nach der `-f` Option ist in dem Beispiel die Gerätedatei für den ersten SCSI-Streamer angegeben worden. Das Gerät wird im "Rewind On Close" Modus betrieben, das Band also nach Beendigung des Kommandos automatisch zurückgespult. Wenn Sie kein Gerät angeben, versucht `tar` auf das bei der Übersetzung des Programms voreingestellte Gerät zuzugreifen. Wenn Sie ein anderes Gerät für mehrere Aufrufe von `tar` oder `mt` voreinstellen möchten, können Sie das über die Umgebungsvariable `TAPE` machen.

Mit der `-g` Option wird eine Datei bestimmt, die gleichzeitig Zeitmarke des Backups und Inhaltsverzeichnis des Archives ist. Hier werden nur die Verzeichnisse, nicht die einzelnen Dateien eingetragen.

Mit Hilfe dieser Datei ist es besonders einfach, inkrementelle Backups zu machen. Wenn Sie beispielsweise an einem anderen Tag alle Veränderungen relativ zu dem vollständigen Backup sichern wollen, können Sie folgendes Kommando eingeben:

```
# export TAPE=/dev/fd0
# tar -c -z -V "Backup Level 1 vom 28.08.94" -g /var/adm/Backup/Dir /
tar: Removing leading / from absolute path names in the archive.
# _
```

Jetzt werden automatisch alle Dateien gespeichert, deren Änderungszeit neuer als das letzte Backup ist. Hierbei werden Dateien, die mit mv umbenannt oder verschoben worden sind ebensowenig gesichert wie zusätzlich installierte, ältere Pakete, deren Erzeugungszeit beim Auspacken normalerweise nicht verändert wird.

Je nachdem wie das Linux-System genutzt wird, reicht für ein inkrementelles Backup auch eine Diskette als Speichermedium aus. In diesem Beispiel werden die Daten komprimiert, so daß mehr als drei Megabyte Textdaten auf einer rohen Diskette gespeichert werden können. Weil ein fehlerhafter Block auf der Diskette das gesamte Archiv ab diesem Block unbrauchbar macht, sollten Sie besonders zur komprimierten Datensicherung nur geprüfte Markendisketten verwenden.

Wenn bei den einzelnen Sicherungsschritten mehr Daten anfallen, als auf einer Diskette Platz finden, werden Sie wahrscheinlich auch für die inkrementellen Backups Magnetbänder verwenden wollen. Um die Kapazität der Bänder auszunutzen, können Sie mehrere separate Archive hintereinander auf einem Band speichern. Zum Positionieren des Bandes müssen Sie das separate Kommando mt benutzen.

```
# export TAPE=/dev/nftape
# mt eom
# tar -c -V "Backup Level 1 vom 30.08.94" -g /var/adm/Backup/Dir /
tar: Removing leading / from absolute path names in the archive.
# mt rewind
```

Mit dem ersten mt-Kommando wird das Band bis an das Ende der zuletzt geschriebenen Daten vorgespult. Dabei ist es gleichgültig, wieviele Dateien/Archive bereits auf das Band geschrieben worden sind.

Weil das Bandgerät im ``No Rewind On Close" Modus betrieben wird (der Modus wurde durch die Gerätedatei gewählt), schreibt das tar-Kommando seine Daten von dieser Stelle an. Um das Magnetband zu schonen, ist es sinnvoll, das Band vor dem Entfernen des Cartridges aus dem Laufwerk zurückzuspulen.

Wenn Sie mehrere Archive auf einem Band gespeichert haben, kommen Sie an ein bestimmtes Archiv nur mit dem mt-Kommando heran. Magnetbänder haben kein Inhaltsverzeichnis, deshalb müssen Sie sich selbst merken, wieviele Archive auf dem Band gespeichert sind. Um beispielsweise das vierte Archiv auf dem Band zu erreichen, müssen Sie drei Archive überspringen:

```
# export TAPE=/dev/nftape
# mt fsf 3
# cd /
# tar -x
# mt rewind
# _
```



Mit der `-x`-Option veranlassen Sie `tar`, alle Dateien aus dem Archiv zu extrahieren. Durch dieses Kommando restaurieren Sie den Zustand des Dateisystems zum Zeitpunkt der Sicherung. Um ein vollständig zerstörtes System zurückzusichern, müssen Sie alle Backups in der Reihenfolge ihres Entstehens auf die leere Festplatte einspielen.

Das `tar`-Programm bietet noch eine Vielzahl weiterer Varianten der Datensicherung, die Sie beispielsweise im TEX-Info Dokument zu `tar` nachlesen können.

## Datensicherung mit `afio` und `cpio`

Das `cpio`-Kommando und die modernere Variante `afio` bilden die Grundlage für ein sehr flexibles Backupsystem. Die Flexibilität wird erreicht, indem sich die beiden Kommandos auf einen Teilaspekt der Sicherungsarbeit konzentrieren: sie erzeugen und verwalten ein Archiv, einen Datenstrom aus bestimmten Dateien. Die Namen der zu archivierenden Dateien müssen den Kommandos im Standardeingabekanal, beispielsweise als Ausgabe von `find`, übergeben werden. Damit sind `cpio` und `afio` zur Verwendung in Shellscripten prädestiniert.

Während `tar` bei den inkrementellen Sicherungen einfache und feste Kriterien zur Unterscheidung alter und neuer Dateien anlegt, kann durch das Zusammenspiel mehrerer auf bestimmte Teilaufgaben spezialisierter Werkzeuge eine sehr genaue Liste der veränderten Dateien erzeugt werden.

`afio` bietet zusätzlich die Möglichkeit, während der Archivierung die einzelnen Dateien zu komprimieren, ohne das gesamte Archiv durch den Kompressor zu leiten. Das hat den Vorteil, daß bei kleineren Fehlern im Archiv nur einzelne Dateien betroffen sind. Außerdem können so komprimierte Archive auf mehrere Bänder verteilt werden.

Wenn Sie die Vorteile von `afio` oder `cpio` nutzen wollen, brauchen Sie sich das dazu notwendige Script nicht selbst zu schreiben. Besonders empfehlenswert ist das `backup-1.03` Paket für `afio` von Karel Kubat, das auf verschiedenen FTP-Sites zu finden ist.

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [Der Druckerdämon lpd](#) **Up:** [Systemverwaltung](#) **Previous:** [Softwaremanagement](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)



## Subsections


- [Den lpd erziehen](#)
    - [/etc/printcap](#)
  - [Die Druckerfilter](#)
    - [Der Ausgabefilter](#)
    - [Die Eingabefilter](#)
- 

# Der Druckerdämon lpd

Bereits bei den urtümlichen Vorfahren unseres heutigen Linux hat sich eine besondere Gruppe von Programmen entwickelt, die in symbiotischer Beziehung zum Betriebssystemkern stehen. Sie arbeiten automatisch und erfüllen immer wiederkehrende Aufgaben. Weil sie vom normalen Benutzer niemals direkt aufgerufen werden, also ihre Arbeit unsichtbar im Hintergrund erledigen, werden sie als Dämonen bezeichnet.

Ein typischer Dämon ist der lpd, der Line Printer Dämon. Dieser Dämon kann die Druckaufträge (Jobs) eines Anwenders entgegennehmen und sie auf einen geeigneten Drucker weiterleiten. Gegenüber dem ebenfalls möglichen Verfahren der direkten Übertragung an einen Drucker, beispielsweise mit dem cat-Kommando, hat die Benutzung des Dämons eine Reihe von Vorteilen.

- Der Dämon kann immer Druckaufträge annehmen, auch wenn kein Drucker im System frei ist (Spooler).
- Mit Hilfe des Druckerdämons kann ein Druckauftrag in einem Netzwerk auf jedem geeigneten angeschlossenen Drucker ausgeführt werden. Insbesondere können beispielsweise auch alle Druckjobs von allen Rechnern eines Netzes auf einem einzigen Drucker zentral ausgeführt werden.
- Ein Druckerdämon kann mit sogenannten Filtern sehr unterschiedliche Druckvorlagen für jeden Drucker korrekt aufbereiten.

Als ordentlicher Dämon tritt der lpd nicht direkt in Erscheinung. Für den Anwender sind die Kommandos lpr, lpq und lprm vorgesehen. 

Mit dem lpr-Kommando können Sie ein Dokument an den Druckerdämon übergeben. Damit wird der im Hintergrund schlafende Dämon aktiviert, der das Dokument in das Spoolverzeichnis kopiert und eine zusätzliche Befehlsdatei schreibt. Auf diese Weise entsteht ein Druckjob, der sich in die Warteschlange (Queue) des Druckerdämons einreihet.

Mit dem Kommando lprm (line printer remove) kann ein abgeschickter Druckjob angehalten werden. Das lpq-Kommando zeigt den aktuellen Status der Druckerwarteschlange, das ist die Liste aller unbearbeiteten Jobs.

Die Systemverwalterin kann mit dem `lpc` (line printer control) Kommando den Status des Druckerdämons überprüfen und verändern.

## Den lpd erziehen

Erwartungsgemäß muß ein Dämon, der seine Arbeit ja ohne direkten Kontakt mit dem Benutzer erledigt, von der Systemverwalterin konfiguriert werden.

Dazu dient die Datei `/etc/printcap`, in der das grundsätzliche Verhalten des Dämons eingestellt werden kann, sowie eine Reihe von Filtern, mit denen der Dämon aus rohen Texten wohlgesetzte Dokumente destilliert.

Wenn der Druckerdämon korrekt läuft, sollte er in der Datei `/etc/rc.local` (oder in einer vergleichbaren Datei) beim Systemstart automatisch aufgerufen werden.

Um den Druckerdämon netzwerkfähig zu machen, müssen alle Rechner, von denen der Zugriff auf den lokalen Drucker erlaubt werden soll, in der Datei `/etc/hosts.equiv` oder `/etc/hosts.lpr` aufgeführt werden.

In einer Datei mit dem Namen `minfree` im Spoolverzeichnis des Dämons kann eine Anzahl Plattenblöcke festgelegt werden, die vom Dämon freigehalten wird.

### `/etc/printcap`

Die `printcap` -Datei beschreibt das Spoolsystem und die Druckerfilter.

Jede Zeile, die nicht von einem `#` eingeleitet wird, definiert einen Drucker im System. Wenn ein Eintrag über mehrere Zeilen gehen soll, muß das Zeilenende durch einen Backslash versteckt werden. Ein Eintrag besteht aus mehreren Feldern, die durch Doppelpunkte voneinander getrennt werden. Das erste Feld enthält den Namen, unter dem der Drucker angesprochen wird. Die anderen Felder belegen Variable oder setzen Schalter zur Einstellung des Druckerdämons. Die Zuweisung an Zeichenvariable erfolgt mit dem Gleichheitszeichen, numerische Variable werden mit dem Nummernzeichen `#` belegt.

Die folgenden Variablen können mit Zeichenketten (Wörter, Namen) belegt werden.

`lp=`

Die Gerätedatei des Druckers. Für Netzwerkdrucker an anderen Hosts muß diese Variable leer definiert werden, damit der voreingestellte Drucker gelöscht wird. Die Voreinstellung ist meistens `/dev/lp`.

`rm=`

Der Name des Remote Host, an dem der Netzwerkdrucker angeschlossen ist.

`rg=`

Die Benutzergruppe, deren Mitglieder den Drucker benutzen dürfen.

`rp=`

Name des Druckers am Remote Host. Voreingestellt ist der Standarddrucker `lp`.


`lo=`

Name der Lockdatei. Voreinstellung ist `lock`. Das Lockfile wird im Spoolverzeichnis des Druckers vom Druckerdämon (Kind) angelegt.

st=

Name des Statusfiles. Voreinstellung ist `status`. Die Statusdatei wird vom Druckerdämon (Kind) im Spoolverzeichnis des Druckers angelegt.

sd=

Der Name des Spoolverzeichnisses. Voreingestellt ist, je nach Distribution, `/var/spool/lpd` oder `/usr/spool/lpd`. 

af=

Name der Abrechnungsdatei. Der Name dieser Datei wird den Eingabefiltern übergeben, die dort Informationen über den Umfang des Druckjobs ablegen sollten, mit denen die Systemverwalterin die Druckkosten mit den Benutzern abrechnen kann.

ff=

Die Zeichenkette für den Seitenvorschub.

tr=

(trailer) Diese Zeichenkette wird nach Beendigung des letzten Druckjobs ausgegeben.

lf=

Name des Logbuchs. In dieser Datei werden die Fehlermeldungen des Druckerdämons gespeichert.

Die folgenden Variablen bestimmen die Filter zur Aufbereitung der zu druckenden Dateien.

of=

Der Name des Ausgabefilters.

if=

Name des (Standard-) Eingabefilters

rf=

Name des FORTRAN-Textfilters. Dieser Filter wird von `lpr` mit der `-r` Option ausgewählt.

tf=

Name des `troff`-Filters. Dieser Filter wird von `lpr` mit der Option `-t` ausgewählt.

nf=

Name des `ditroff`-Filters. Dieser Filter wird von `lpr` mit der Option `-n` ausgewählt.

df=

Name des TEX-Filters. Dieser Filter wird von `lpr` mit der Option `-d` ausgewählt.

gf=

Name des `graph`-Filters. Dieser Filter wird von `lpr` mit der Option `-g` ausgewählt.

vf=

Name des Rastergrafik-Filters. Dieser Filter wird von `lpr` mit der Option `-v` ausgewählt.

cf=

Name des `cifplot`-Filters. Dieser Filter wird von `lpr` mit der Option `-c` ausgewählt.

Die folgenden Variablen können mit numerischen Werten (Zahlen) belegt werden.

mx#

Die maximale Blockzahl (Größe) eines Druckjobs. Durch die Zuweisung `mx#0` wird die Jobgrößenbeschränkung abgeschaltet.

mc#  
Die maximale Anzahl erlaubter Kopien eines Druckjobs.

pw#  
Die (maximale) Spaltenzahl einer Druckseite.

px#  
Die Breite einer Druckseite in Pixeln.

pl#  
Die (maximale) Zeilenzahl einer Druckseite.

py#  
Die Höhe einer Druckseite in Pixeln.

Die folgenden Schalter verändern das Verhalten des Druckerdämons.

rs  
Der Zugriff auf einen Netzwerkdrucker ist nur denjenigen Anwendern gestattet, die einen Account auf dem Host mit dem Drucker haben.

sc  
Verbietet mehrfachen Ausdruck eines Dokumentes.

sf  
Unterdrückt die Ausgabe eines Seitenvorschubs nach jedem Druckjob.

sh  
Unterdrückt die Ausgabe einer Titelseite vor jedem Druckjob.

sb  
Verkürzt die Titelseite auf eine Zeile.

hl  
Veranlaßt den Druckerdämon, die Titelseite nach jedem Druckjob zu schreiben.

rw  
Öffnet den Drucker zum Lesen und Schreiben.

Die folgenden numerischen Variablen stellen die Schnittstelle für serielle Drucker ein.

br#  
Die Baudrate, wenn der Drucker an einer seriellen Schnittstelle hängt.

fc#  
Löscht die im Argument gesetzten Bits.

fs#  
Setzt die im Argument gesetzten Bits.

xc#  
Löscht die im Argument gesetzten local-mode-Bits.

xs#  
Setzt die im Argument gesetzten local-mode-Bits.

Ein typischer Eintrag sieht beispielsweise so aus:

```
# Standarddrucker an lp1 mit Spoolverzeichnis lp1 und Filter lpf
lp:lp=/dev/lp1:sd=/var/spool/lp1:of=/usr/lib/lpf:lf=/var/adm/lp:mx#0
```

In diesem Beispiel wird ein Drucker an der ersten parallelen Schnittstelle `/dev/lp1` unter dem Namen `lp` definiert. Als Spoolverzeichnis (`sp`) wird `/var/spool/lp1` verwendet, als Ausgabefilter (`of`) wird das Programm `lpf` festgelegt, durch die Zuweisung des Wertes 0 an die Variable `mx` werden beliebig große Dokumente zugelassen, und als Logfile wird die Datei `/usr/adm/lp` bestimmt. Vorausgesetzt, die genannten Verzeichnisse und Dateien existieren, sollte mit diesem Eintrag der erste Probeausdruck möglich sein.

Eine kleine Veränderung des Beispieleintrages veranlaßt den Druckerdämon dazu, das Dokument über das TCP/IP-Netzwerk an einen anderen Rechner zu schicken und von dem dortigen Druckerdämon verarbeiten zu lassen:

```
# Standarddrucker ueber Netz an Rechner soho
lp:lp=:rm=soho:lf=/usr/adm/lpd-errs:
```

## Die Druckerfilter

Mit Ausnahme des Papierformates sind in der `printcap`-Datei keine Informationen über die genaue Ansteuerung des Druckers gespeichert. Das bedeutet, daß der Druckerdämon nicht einmal einen Reset oder eine Druckerinitialisierung selbst ausführen kann. Trotzdem kann ein gut eingerichteter `lpd` eine Vielzahl unterschiedlicher Formate auf jedem angeschlossenen Drucker ausgeben, vorausgesetzt, die Systemverwalterin hat die passenden Filter installiert.

Der `lpd` kann einen Ausgabefilter (`of`) und bis zu neun Eingabefilter (`cf`, `df`, `gf`, `if`, `nf`, `rf`, `tf`, `vf` und `pr`, siehe oben) benutzen.

## Der Ausgabefilter

Der Ausgabefilter ist in erster Linie dazu gedacht, die Ausgabe der vom `lpd` selbst generierten Titelseiten am Beginn jedes Druckjobs zu filtern. Weil eine eventuell notwendige Druckerinitialisierung bereits für diese Titelseite sinnvoll ist, sollte der Ausgabefilter auch diese Aufgabe erfüllen.

Das Zusammenspiel des Ausgabefilters mit dem `lpd` ist insofern etwas kompliziert, als dieser nur einmal zu Beginn des ersten Druckjobs gestartet wird. Für den Ausdruck der eigentlichen Dokumente benutzt der Druckerdämon einen der Eingabefilter und hält den Ausgabefilter bis zum Beginn des nächsten Jobs an. Dazu schreibt der Dämon die Zeichenkette `\031\1` in die Standardeingabe des Ausgabefilters. Der Filter muß sich daraufhin selbst mit `SIGSTOP` anhalten und auf das `SIGCONT`-Signal warten.

## Die Eingabefilter

Die Zahl möglicher Formate, in denen ein Dokument vorliegen kann, und die Zahl möglicher Druckersprachen, in die dieses Format umgewandelt werden muß, um tatsächlich auf dem Papier zu erscheinen, ist so groß, daß es keine einfache Methode gibt, diese Übersetzung direkt vom Druckerdämon ausführen zu lassen.

Deshalb muß die Systemverwalterin dem `lpd` für jedes Eingabeformat einen Eingabefilter bereitstellen, der dieses Format für den installierten Drucker aufbereitet.

Das Zusammenwirken des `lpd` mit den Eingabefiltern ist sehr einfach. Mit einer der Optionen `c`, `d`, `f`, `g`, `n`, `p`, `t` und `v` gibt der Benutzer beim Aufruf von `lpr` an, welches Format sein Dokument hat, und bestimmt so, welcher Eingabefilter benutzt wird. Dieser Filter wird dann vom `lpd` für den Ausdruck des bei diesem Aufruf von `lpr` bestimmten Dokumentes in einer Pipeline gestartet. Der Dämon ruft den `if`-Filter mit folgender Kommandozeile auf:

```
if [-c] -wSpalten -lZeilen -i Einrückung -n Name -h Host
Accountdatei
```

Die `-c` Option ist gesetzt, wenn der `lpr` mit der Option `-l` aufgerufen wurde. Alle anderen Eingabefilter werden mit der Zeile

```
filter -x Breite -y Länge -n Name -h Host Accountdatei
```

aufgerufen.

Während ein Eingabefilter aktiv ist, bleibt der Ausgabefilter angehalten. Der Eingabefilter liest aus der Pipeline (Standardeingabe), verändert den Datenstrom seinem Programm entsprechend und schreibt die Daten in die Standardausgabe, die direkt mit dem Drucker verbunden ist.

So ein Filter kann ein vollständiges C-Programm sein. Das dem `lpd`-Paket beiliegende Programm `lpf` ist beispielsweise ein Eingabefilter speziell für `groff`-Dokumente.

Als Eingabefilter können aber auch Shellscripts oder `perl`-Programme verwendet werden. Auf diese Weise kann zum Beispiel die Umwandlung einer Postscriptdatei für einen Epson-Nadeldrucker mit dem Ghostscript-Interpreter `gs` erfolgen:

```
#!/bin/bash
#
gs -q -dNOPAUSE -sDEVICE=EPSON -sOutputFile=- -
```

Ein schönes Beispiel für die vielfältigen Möglichkeiten, einen Filter zu bauen, ist das `magicfilter` von Peter Anvin. Dieses Shellprogramm benutzt "Magische Zahlen" zur Erkennung der Dateiformate, ähnlich wie `file`. Der Magische Filter koordiniert die Zusammenarbeit verschiedener Programme um eine Datei möglichst optimal für einen bestimmten Drucker aufzubereiten. Sie finden die Sourcen zu diesem Filter im Internet, beispielsweise auf allen Mirrors von `ftp://sunsite.unc.edu` im Verzeichnis `Linux/system/Printing`.

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [Der Batchdämon crond](#) **Up:** [Systemverwaltung](#) **Previous:** [Datensicherung](#)

## Subsections

- [Der Terminkalender crontab](#)
    - [Die crontab-Datei](#)
    - [Beispiele:](#)
    - [Umgebungsvariable in der crontab-Datei](#)
    - [Das crontab-Kommando](#)
- 

# Der Batchdämon crond

Der vielseitigste aller Dämonen ist der `crond` (gesprochen cron-d). Er führt beliebige Kommandos automatisch zu vorbestimmten Zeitpunkten aus, wie nach einem Fahrplan. Einmal pro Minute sieht der Dämon in seinen Terminkalender und führt zuverlässig und pünktlich alle anstehenden Kommandos aus.

Der unter Linux verbreitete `crond` von Paul Vixie (`vixie-cron`) erlaubt prinzipiell allen Benutzerinnen, mit dem `crontab`-Kommando eine eigene Spalte in dem Terminkalender des Dämons einzurichten. Die in dieser Spalte eingetragenen Befehle werden dann „im Auftrag“, also mit der Benutzerkennung der Auftraggeberin, ausgeführt.

Wie der Druckerdämon `lpd` wird der `crond` normalerweise während der Systeminitialisierung aus einer der `rc*`-Dateien aufgerufen. Der Dämon geht von selbst in den Hintergrund.

Der Terminkalender des Dämons befindet sich im Verzeichnis `/var/spool/cron/crontabs` und besteht aus beliebig vielen Spalten, jede in Form einer Datei mit dem Namen der Auftraggeberin.

## Der Terminkalender crontab

Die einzelnen Dateien für den Terminkalender des Dämons werden von den Auftraggeberinnen verwaltet. Es handelt sich dabei um einfache ASCII-Textdateien, die beispielsweise mit dem `elvis`-Editor erzeugt werden können. Weil das `/crontabs`-Verzeichnis des Dämons für normalsterbliche Systembenutzerinnen nicht beschreibbar ist (es ist normalerweise nicht einmal lesbar), muß eine Vorlage für die Termindatei in einem anderen, beschreibbaren Verzeichnis angelegt werden. Die Veränderung des `/crontabs`-Verzeichnisses wird dann vom `crontab`-Kommando mit Superuser-Rechten vorgenommen.

## Die crontab-Datei

In einer Termindatei werden alle Zeilen, die weder leer sind noch mit einem `#`-Zeichen in der ersten Spalte als Kommentar markiert sind, vom Dämon bearbeitet. Solche Zeilen können entweder eine Umgebungsvariable für die Ausführung aller in dieser Datei aufgerufenen Kommandos definieren

oder eine Zeitmaske mit zugehörigen Kommando enthalten.

Eine Zeitmaske besteht aus fünf Feldern, die durch Leerzeichen voneinander getrennt werden.

Die Zeitmaske

Feldnummer	1	2	3	4	5
Bedeutung	Minute	Stunde	Monatstag	Monat	Wochentag
Bereich	0-59	0-23	0-31	0-12*	0-7*
6r* oder Namen					

Die Monate und Wochentage können auch mit ihren englischen Namen angegeben werden. Die Namen können auf drei Zeichen abgekürzt werden, Groß-/Kleinschreibung wird ignoriert.

Bei Zahlendarstellung des Wochentages entsprechen 0 und 7 dem Sonntag.

Jedes Feld der Zeitmaske kann durch einen Asterisk '\*' belegt werden, der auf jeden Termin paßt.

Die Felder können durch Komma getrennte Listen von Zeiteinträgen sowie Bereiche der Form *von-bis* enthalten.

In Bereichen dürfen keine Namen verwendet werden. Es können auch mehrere Bereiche aufgelistet werden. Zusätzlich können den Bereichen noch „Teiler“ zur Veränderung der Schrittweite nachgestellt werden.

Die restliche Zeile bis zum Zeilenende oder einem %-Zeichen wird als Kommando ausgeführt. Wenn ein %-Zeichen gefunden wird, das nicht durch einen Backslash entwertet ist, wird der Rest der Zeile als Eingabe an das Kommando geleitet.

## Beispiele:

```
0 8-18 0 * 1-5 /usr/lib/newsbin/input/newsrun
```

Das newsrun-Kommando wird montags bis freitags von 8 bis 18 Uhr zu jeder vollen Stunde aufgerufen.

```
0 17 24 12 * echo %schoene Bescherung
```

Heiligabend um 17 Uhr, egal was für ein Wochentag. Die Mitteilung wird per Mail an den Auftraggeber geschickt. In diesem Beispiel wird der Mitteilungstext nicht in der Kommandozeile an das echo-Kommando übergeben, sondern in seinen Standardeingabekanal geschrieben.

```
0-59/5 * * * ~/bin/remind
```

Das remind-Kommando wird alle fünf Minuten aufgerufen, jeden Monat, jeden Tag, jede Stunde ...

```
0 6,10,14,18,22 * * * /usr/lib/uucp/uucico -s uucphost
```

Das uucico-Kommando wird täglich um 6, 10, 14, 18 und 22 Uhr auf- und damit der uucphost angerufen.

Ein bestimmter Tag kann sowohl als Monatstag als auch als Wochentag bestimmt werden. Wenn beide Felder bestimmt, also nicht durch einen Asterisk besetzt sind, werden sie durch eine logische ODER-Verknüpfung ausgewertet. Eine Warnung der Form

```
0 0-23 13 * fri wall %Achtung, Freitag der 13. *** FALSCH ***
```



würde also jeden Freitag und jeden 13. eines Monats ausgegeben.

## Umgebungsvariable in der crontab-Datei

Wie bereits erwähnt, können in der `crontab`-Datei auch Umgebungsvariable für die Ausführung der Kommandos bestimmt werden. Die Variablen `HOME`, `LOGNAME` und `SHELL` werden von `crond` automatisch mit den Werten aus der `passwd`-Datei vorbelegt. Die Variable `LOGNAME` kann nicht verändert werden.

In der `MAILTO`-Variablen kann ein realer Account bestimmt werden, an den die Ausgabe der automatischen Kommandos geschickt wird. Wenn die Variable definiert, aber leer ist, geht die Ausgabe der Kommandos verloren. Wenn die Variable nicht definiert ist, wird die Ausgabe automatisch an den Eigentümer der Terminkalenderspalte gesendet.

Die Definition von Umgebungsvariablen erfolgt wie in normalen Shellscripts durch Zuweisung einer (möglicherweise leeren) Zeichenkette.

## Das crontab-Kommando

Mit dem Anwenderkommando `crontab` werden die benutzerdefinierten Termindateien im Verzeichnis `/var/spool/cron/crontabs` verwaltet.

`crontab` erkennt folgende Optionen:

### `-u Benutzer`

(user) legt den Benutzernamen fest, in dessen Auftrag eine Termindatei ausgeführt werden soll. Diese Option ist nur dem Superuser (root) zugänglich. Alle normalsterblichen Systembenutzer können (höchstens) ihre eigenen Termindateien bearbeiten.

### `-l`

(list) zeigt den Inhalt der aktuellen Termindatei an.

### `-d`


(delete) löscht die Termindatei aus dem Verzeichnis `./crontabs`.

### `-r Datei`

(replace) ersetzt die Termindatei im `./crontabs`-Verzeichnis durch die angegebene.

Wenn im Verzeichnis `/var/spool/cron` die Datei `allow` existiert, kann das `crontab` nur von den darin aufgeführten Systembenutzern ausgeführt werden. Wenn anstelle der `allow`-Datei eine Datei mit dem Namen `deny` existiert, steht das `crontab`-Kommando allen Systembenutzern zur Verfügung, die NICHT darin aufgeführt sind.

Wenn keine der Dateien existiert, hängt das Verhalten von `crontab` von den Einstellungen bei der Compilierung ab. Entweder kann jeder das `crontab`-Kommando ausführen, oder nur der Superuser (root).

Wenn das `./crontabs`-Verzeichnis vom `crontab`-Kommando verändert wird, liest der `crond` automatisch die neuen Daten, muß also nicht extra neu gestartet werden .

**Next:** [Der Protokollschreiber syslogd](#) **Up:** [Systemverwaltung](#) **Previous:** [Der Druckerdämon lpd](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

**Next:** [Recompilieren des Kernels](#) **Up:** [Systemverwaltung](#) **Previous:** [Der Batchdämon crond](#)

## Subsections

- [Die Datei /etc/syslog.conf](#)
  - [Beispiele:](#)
- 

# Der Protokollschreiber syslogd

Das Betriebssystem (der Kernel) und seine Prozesse auf der Benutzerebene (Dämonen, Kinder von `init`) sind mit keinem Benutzerterminal direkt verbunden. Diese Selbständigkeit wird dann zu einem Problem, wenn ein solcher Prozeß eine Nachricht, beispielsweise eine Fehlermeldung, ausgeben will/muß. Die Kanäle für die Standardausgabe und die Standardfehlerausgabe sind bei den Systemprozessen mit der Gerätedatei `/dev/console` verbunden. Die Ausgabe für dieses Gerät wird auf dem Bildschirm der Systemconsole direkt ausgegeben; unter X11 kann sie im `xconsole`-Fenster angezeigt werden.

Im Mehrbenutzerbetrieb ist die Methode, eine Systemmeldung ausschließlich als Fehlermeldung auf die Systemconsole zu schreiben, unbefriedigend. Es kann nicht sichergestellt werden, daß die Meldung von der „richtigen“ Person gelesen wird. Die Bildschirmmeldungen lassen sich nicht sichern und gehen so extrem leicht verloren.

Eine sehr umfassende Lösung dieses Problems bietet der `syslogd` (gesprochen süslog-d) an. Allerdings muß dieser Lösungsweg bereits bei der Programmierung des Systemprogramms eingeschlagen werden. Wegen der eindeutigen Vorteile arbeitet bereits ein großer Teil der Linux-Systemprogramme mit dem `syslogd` zusammen.

Anstelle der direkten Ausgabe einer Meldung mit `fprintf(3)` oder einer vergleichbaren C-Bibliotheksfunktion können Systemmeldungen von spezieller Bedeutung mit der `syslog(3)`-Funktion ausgegeben werden, die Bestandteil der Standardbibliothek des `gcc` für Linux ist.

Solche Meldungen werden dann automatisch vom `syslogd` entgegengenommen. 

Der `syslogd` erhält die Nachrichten über das Socket `/dev/log` und über die Spezialdatei `/proc/kmsg`. Er liefert die Meldung nach den Regeln aus, die die Systemverwalterin in der Konfigurationsdatei `/etc/syslog.conf` festgelegt hat.

## Die Datei /etc/syslog.conf

Der Dreh- und Angelpunkt der `syslog`-Installation ist die Datei `/etc/syslog.conf`. In dieser Konfigurationsdatei werden für verschiedene Klassen von Nachrichten verschiedene Auslieferungswege bestimmt.

Zur Klassifizierung gibt die Systemprogrammiererin jeder Syslognachricht zwei Merkmale: Herkunft und Priorität.

Für jede Herkunftskategorie können Nachrichten ab einer bestimmten Priorität auf einen Auslieferungsweg geschickt werden.

Als Herkunft kommen folgende Bereiche in Frage:

kern  
für die Systemmeldungen direkt aus dem Betriebssystemkern.

auth  
für Meldungen der Sicherheitsdienste (login u. ä.).

authpriv  
für vertrauliche Meldungen der internen Sicherheitsdienste.

mail  
für Meldungen des Mailsystems.

news  
für Meldungen des Nachrichtendienstes.

uucp  
für Meldungen des Fernkopierers.

lpr  
für Meldungen des Druckerdämons.

cron  
für Meldungen des crond.

syslog  
für Meldungen vom syslogd selbst.

daemon  
für Meldungen des unbekannten Dämons.

user  
für Meldungen aus normalen Anwenderprogrammen.

local0-7  
frei verwendbar für Nachrichten lokaler Bereiche.

Die Prioritäten (von der höchsten zur niedrigsten):

emerg  
für den letzten Spruch vor dem Absturz.

alert  
für alarmierende Nachrichten, die sofortiges Eingreifen erfordern.

crit  
für die Meldung kritischer Situationen, die gerade noch gut gegangen sind.

err  
für Fehlermeldungen aller Art.

warning  
für Warnungen vor Hunden, Käfern und anderen Gefahren.

notice  
zur Dokumentation besonders bemerkenswerter Situationen im Rahmen des normalen Betriebs.

info  
für die Protokollierung des normalen Betriebsgeschehens.

debug

zur Mitteilung innerer Programmzustände bei der Fehlersuche.

none

ist keine Priorität im eigentlichen Sinne, sondern dient dazu, alle Nachrichten einer Herkunftskategorie von einem Transportweg auszuschließen.

Die Nachrichten können auf vier verschiedene Weisen ausgeliefert werden:

1.

können sie in eine Datei geschrieben werden. Dazu wird der Dateiname mit absolutem Pfad angegeben (mit einem Slash '/' beginnend).

2.

können sie an den `syslogd` eines anderen Rechners im lokalen Netz weitergeleitet werden. Dazu wird der Zielrechnername, eingeleitet von einem '@'-Zeichen, angegeben.

3.

können die Nachrichten an bestimmte Systembenutzer geschickt werden. Die Benutzernamen müssen dazu in einer durch Komma getrennten Liste aufgeführt werden. Die Nachricht wird den aufgelisteten Benutzern angezeigt, sofern diese im Augenblick des die Nachricht auslösenden Ereignisses eingeloggt sind.

4.

können sie an alle eingeloggten User geschickt werden, indem ein Asterisk '\*' anstelle des Usernamens angegeben wird.

Ein Eintrag in der `syslog.conf` besteht aus einer Zeile mit einem *Herkunft.Priorität*-Paar (durch einen Punkt voneinander getrennt) und einer Wegdefinition. In einem *Herkunft.Priorität*-Paar können auch mehrere Herkunftskategorien durch Komma getrennt aufgelistet werden. Außerdem können mehrere *Herkunft.Priorität*-Paare durch Semikolon getrennt in einer Zeile für einen Weg angegeben werden. Sowohl *Herkunft* als auch *Priorität* können durch einen Asterisk '\*' als Wildcard ersetzt werden.

Kommentare können mit dem '#' eingeleitet werden.

## Beispiele:

```
# Alle Kernelmessages, sowie die Nachrichten der Sicherheitsdienste
# mit der Priorität notice oder höher und alle Nachrichten mit
# der Priorität err und höher werden auf der Systemconsole
# angezeigt.
# Die vertraulichen Nachrichten des Sicherheitsdienstes werden von
# dieser Anzeige ausgeschlossen.
```

```
kern.*;auth.notice;*.err;authpriv.none /dev/console
```

```
# Alle Nachrichten mit Priorität info und höher, außer den Nachrichten
# des Mailsystems und des privaten Sicherheitsdienstes, werden in der
# Datei /usr/adm/Logbuch mitgeschrieben.
```

```
*.info;mail.none;authpriv.none /usr/adm/Logbuch
```

```
# Die Nachrichten des privaten Sicherheitsdienstes kommen in eine
```

```
# sichere Datei.

authpriv.*                                /home/she/privat/.sicher/Nachrichten

# Alle Nachrichten mit Priorität emerg werden allen eingeloggten
# Usern sofort angezeigt, und sie werden an den syslogd eines anderen
# Rechners weitergeleitet.

*.emerg                                     *
*.emerg                                     @soho.lunetix.de

# Die Nachrichten der Priorität alert und höher werden den
# verantwortlichen Systemadministratoren angezeigt, wenn sie gerade
# eingeloggt sind.

*.alert                                     root,she

# Die kritischen Fehlermeldungen des News- und Mailsystems werden in
# der Datei /usr/adm/Kommunikationsstoerung gespeichert

uucp,mail,news.crit                       /usr/adm/Kommunikationsstoerung
```

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [Recompilieren des Kernels](#) **Up:** [Systemverwaltung](#) **Previous:** [Der Batchdämon crond](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

## Subsections

- [Entpacken der Quelltexte](#)
  - [Die Kernel-Konfiguration](#)
  - [make xconfig](#)
  - [make menuconfig](#)
  - [Die Konfigurationsmöglichkeiten im Einzelnen](#)
    - [Code maturity level options](#)
    - [Loadable module support](#)
    - [General setup](#)
    - [Floppy, IDE and other block devices](#)
    - [Networking options](#)
    - [SCSI support](#)
    - [SCSI low-level drivers](#)
    - [Network device support](#)
    - [ISDN subsystem](#)
    - [CDROM drivers \(not for SCSI or IDE/ATAPI drives\)](#)
    - [Filesystems](#)
    - [Charakter devices](#)
    - [Sound](#)
    - [Kernel hacking](#)
- 

# Recompilieren des Kernels

Wie die meisten modernen Betriebssysteme verfügt Linux über eine umfangreiche Auswahl an Treibern für die verschiedensten Hardwaregeräte. Einige dieser Erweiterungskarten besitzen ein eigenes BIOS, das bereits einen Treiber für die entsprechende Karte enthält. Linux verwendet dieses BIOS allerdings nur, um Konfigurationsparameter auszulesen oder das Vorhandensein der Karte überhaupt festzustellen, da die meisten dieser BIOSe in Multitaskingumgebungen, wie Linux (oder OS/2, Windows NT oder ein anderes UNIX) sie zur Verfügung stellt, nicht verwendbar sind. Deshalb muß der Kernel Quelltext vor dem Übersetzen auf die entsprechende Treiberauswahl, die zum Starten des Systems notwendig ist, konfiguriert werden.

Die meisten im Kernel Quelltext enthaltenen Treiber sind auch als sogenannte „Module“ übersetzbar, die zur Laufzeit in den Kernel geladen werden können. Lediglich die Treiber, die beim Systemstart erforderlich sind, müssen fest in den Kernel einkompiliert werden. Diese Treiber sind im wesentlichen das Dateisystem, mit dem die Root-Partition formatiert wurde, und der Festplattentreiber, der die

Platte steuert, auf der sich das Root-Dateisystem befindet. Wird das System komplett via Netzwerk betrieben, müssen nur der Netzwerktreiber und das NFS-Dateisystem in den Kernel eingebunden werden. Die meisten der heutigen Distributionen verwenden bereits bei der Installation modularisierte Kernel, wodurch die Notwendigkeit, eigene, auf das System zugeschnittene Kernel zu bauen, deutlich geringer wurde. Soll der Kernel aber aktualisiert oder neue Hardware in das System eingebaut werden, muß der Kernel meistens umkonfiguriert werden. Ebenso kann es aufgrund der Arbeitsspeichersituation, aus geschwindigkeitstechnischen Gründen oder wegen Hardwarekonflikten sinnvoll sein, einen auf die eigenen Bedürfnisse zugeschnittenen Kernel zu installieren.

## Entpacken der Quelltexte

Der Quellcode des Kernels sollte sich in dem Verzeichnis `/usr/src/linux` befinden. Sind die Sourcen noch nicht entpackt, so müssen sie in dem Verzeichnis `/usr/src` mit dem Kommando `tar xzf Dateiname.tar.gz` entpackt werden. Damit die zu dem jeweiligen Kernel gehörenden Headerdateien beim Compilieren verwendet werden, müssen sich in dem Verzeichnis `/usr/include` Symlinks auf die Verzeichnisse `/usr/src/linux/include/linux` und `/usr/src/linux/include/asm` befinden. Sie werden mit den Kommandos

```
rm -f /usr/include/linux /usr/include/asm
ln -s /usr/src/linux/include/asm /usr/include/asm
ln -s /usr/src/linux/include/linux /usr/include/linux
```

erzeugt. Da die Headerdateien nicht nur zum Übersetzen des Kernels benötigt werden, sind diese Symlinks für alle Programme, die Datenstrukturen des Kernels verwenden, notwendig. Sind mehrere Kernelquelltexte auf der Platte verfügbar, so ist darauf zu achten, daß die Symlinks auf die richtigen Headerdateien zeigen.

## Die Kernel-Konfiguration

Zur Konfiguration des Kernels stehen zur Zeit drei Möglichkeiten zur Auswahl. Alle drei werden über sog. „make targets“ im Kernel Quelltextverzeichnis (normalerweise `/usr/src/linux`) gestartet und erstellen die Konfigurationsdatei `.config`, aus der zur Zeit der Übersetzung die eingestellten Parameter gelesen werden. Die traditionelle Variante ist das bekannte `make config`, das ein Shell-Script startet und linear alle Konfigurationsparameter abfragt, ohne die Möglichkeit zu lassen, innerhalb der Fragen vorwärts oder rückwärts gehen zu können. Ebenso müssen immer alle Fragen bearbeitet werden, selbst wenn nur eine einzige Einstellung geändert werden soll.

### **make xconfig**

Benutzern des X-Window-Systems sei das Kommando `make xconfig` ans Herz gelegt. Es setzt allerdings voraus, daß TCL/TK auf dem System installiert sind.

`make xconfig` startet ein sehr übersichtlich gestaltetes TCL/TK Programm und präsentiert dem Benutzer die systematisch sortierten Konfigurationsbereiche des Linuxkernels in einem Fenster. Wählt man den ersten Menüpunkt im Hauptmenü aus, wird ein weiteres Fenster erzeugt, in dem die einzelnen Parameter über Radiobuttons gesetzt werden können, wobei „y“ für in den Kernel



einbinden steht, „m“ übersetzt die Option als ladbares Modul und „n“ läßt den Treiber ganz weg. Am unteren Rand jedes dieser „Unterfenster“ besteht die Möglichkeit, zum vorhergehenden oder nächsten Fenster umzuschalten oder in das Hauptmenü zurückzukehren. Im Hauptmenü kann die eingestellte Konfiguration mit „Save and Exit“ bestätigt oder über die Menüpunkte „Load Configuration from File“ und „Save Configuration to File“ in einer eigenen Datei gesichert werden, was sich als ausgesprochen praktisch erweist, wenn auf einem Rechner Kernel für verschiedene Systeme erstellt werden sollen. Das Programm verlangt daraufhin die Eingabe eines Dateinamens, unter dem die eingestellte Konfiguration gespeichert wird. Über den Menüpunkt „Quit Without Saving“ kann das Programm ohne Änderungen an der aktuellen Konfiguration verlassen werden. Zu allen Optionen ist unter dem Help-Button kontextsensitive englischsprachige Hilfe verfügbar.

## make menuconfig

Im Prinzip funktioniert auch `make menuconfig` wie `xconfig`, nur daß es im Textmodus arbeitet. Die Konfiguration via `menuconfig` basiert auf der `ncurses`-Bibliothek zur Terminalsteuerung.

Innerhalb des Konfigurationsprogramms werden die Funktionen der unteren Menüleiste mit den Cursortasten  $\leftarrow$  und  $\rightarrow$  sowie der TAB-Taste ausgewählt und mit ENTER bestätigt. Innerhalb des Scroll-Fensters kann der Auswahlbalken mit den Cursor-Tasten  $\uparrow$  und  $\downarrow$  sowie den + und - Tasten bewegt werden. Menüeinträge, bei denen „-->“ am Ende angezeigt wird, schalten in eine Unterauswahl mit weiteren Konfigurationsmöglichkeiten. Bei der Auswahl der Treiber bedeutet „[ ]“, daß der Treiber nur in den Kernel einkompiliert werden kann. Mit „< >“ gekennzeichnete Treiber können sowohl als Module übersetzt als auch fest in den Kernel eingebunden werden. Ist die Position zwischen den Klammern leer, wird der Treiber nicht übersetzt. Befindet sich ein „y“ zwischen den Klammern, wird der Treiber in den Kernel eingebaut, bei „m“ wird er als Modul übersetzt. Die Auswahl wird über die Tasten „y“ (in den Kernel), „m“ (als Modul) oder „n“ (weglassen) gesteuert. Der Menüpunkt „Exit“ oder die Esc-Taste springen in die nächsthöhere Menüebene zurück. Über den Menüpunkt „Save Configuration to an Alternate File“ im Hauptmenü kann die aktuelle Konfiguration in eine Datei gespeichert und mit „Load an Alternate Configuration File“ auch wieder eingelesen werden. Mit Esc Esc oder den Menüpunkt „Exit“ wird das Programm beendet und der Benutzer gefragt, ob er die aktuelle Konfiguration übernehmen möchte. Ebenso wie bei `xconfig` sind über die Menüauswahl HELP oder die Taste „?“ kontextsensitive englischsprachige Hilfstexte abrufbar.

## Die Konfigurationsmöglichkeiten im Einzelnen

Der folgende Abschnitt beschreibt die möglichen Konfigurationsparameter in der Reihenfolge, wie sie in den Menüs des Kommandos `make xconfig` erscheinen. Die Optionsnamen unterscheiden sich leicht in den verschiedenen Konfigurationsvarianten, es sollte aber aus dem Kontext ersichtlich sein, um welche Option es sich jeweils handelt.

Der Linuxkernel für die Intel-PC Architektur besitzt in der aktuellen Version (2.0.14) über 340 Konfigurationsoptionen. Aus Platzgründen sind in diesem Abschnitt deshalb nur jene erwähnt, die grundlegenden Einfluß auf das Kernelverhalten haben. Genauere Informationen zu den einzelnen Treibern sind der englischsprachigen Online-Hilfe und weiterführenden englischen Texten im Verzeichnis `Documentation/` im Linuxquelltextverzeichnis zu entnehmen.

# Code maturity level options

## Prompt for development and/or incomplete code/drivers

bestimmt, ob in den nachfolgenden Menüs auch Treiber und Funktionen selektiert werden können, die sich noch im Entwicklungszustand befinden, nicht vollständig sind oder nicht stabil funktionieren. Diese Option sollte nur ausgewählt werden, wenn kein Wert auf hohe Stabilität gelegt oder ein Treiber unbedingt benötigt wird.

## Loadable module support


### Enable loadable module support

kompiliert in den Kernel Unterstützung für zur Laufzeit ladbare Treiber ein. Als zur Laufzeit ladbare Module lassen sich alle Dateisysteme und fast alle Gerätetreiber übersetzen. Diese Option ist in den meisten Fällen sinnvoll, da selten benötigte Treiber als Module übersetzt werden sollten, damit sie nicht immer Speicher belegen und die Wahrscheinlichkeit von Hardwarekonflikten reduziert wird.

### Set version information on all symbols for modules

ermöglicht die Verwendung der Module auch in älteren oder neueren Kernen. Normalerweise müssen alle Module unter dem Kernel übersetzt werden, unter dem sie auch verwendet werden sollen. Diese Option setzt zu jedem Kernsymbol, das im Modul verwendet wird, eine Versionsnummer, die nur dann geändert wird, wenn sich die entsprechende Kernelfunktion ändert. Auf diese Weise kann beim Laden des Moduls überprüft werden, ob das Modul zum gerade verwendeten Kernel kompatibel ist. Werden Module verwendet, die nicht Teil der Kernelquelltexte sind oder nur in Binärform vorliegen, ist diese Option besonders wichtig.

### Kernel daemon support (e. g. autoload of modules)

compiliert Unterstützung für den sog. „kerneld“ (→ Seite ) in den Kernel ein. Der „kerneld“ ist ein Benutzerebenenprogramm, das bei Bedarf Dateisystemmodule, Treiber, Bildschirmschoner o. ä. laden kann. Es ist im `modules-1.2.8`-Paket und allen neueren Versionen des Pakets enthalten.

## General setup

### Kernel math emulation

muß verwendet werden, falls der Kernel auf Rechnern ohne mathematischen Koprozessor verwendet werden soll. Die Koprozessoremulaton vergrößert den Kernel stark, wird aber abgeschaltet, wenn der Kernel beim Start einen Koprozessor findet.

### Networking support

sollte immer in den Kernel eingebunden werden, da etliche Programme die Funktionalität verwenden, selbst wenn der Rechner keine Verbindung zu einem Netzwerk besitzt.

### Limit memory to low 16MB

sollte in den meisten Fällen nicht notwendig sein. Die Option ist für alte Rechner gedacht, die Cache- oder DMA-Probleme mit mehr als 16MB RAM haben.

### PCI bios support

und

### PCI bridge optimization (experimental)

sollten auf allen neueren PCI Rechnern eingeschaltet werden. Lediglich einige der ersten PCI-Systeme haben BIOS- oder Chipsatzfehler, die eine Abschaltung dieser Optionen erfordern.

### **System V IPC**

ermöglicht Interprozeßkommunikation, eine kernelgestützte Kommunikationsschnittstelle zum Datenaustausch der Programme untereinander. Da viele Programme, insbesondere der Dosemulator und einige Datenbanken, dieses Feature benötigen, sollte es normalerweise eingeschaltet sein.

### **Kernel support for a.out binaries**

Das a.out-Binärformat ist ein altes UNIX-Format für ausführbare Programme. Neuere Linux-Distributionen verwenden das ELF-Format (Executable and Linkable Format). Viele Programme liegen aber noch im a.out-Format vor, so daß auch diese Option eingeschaltet werden sollte.

### **Kernel support for ELF binaries**

Das ELF-Format ist ein (unter Linux) neues Format für Programme und Bibliotheken, das immer häufiger Verwendung findet. Auch diese Option sollte eingeschaltet werden.

### **Kernel support for JAVA binaries**

ermöglicht das Starten von Java-Applets aus der Kommandozeile. Aufgrund der noch recht schwachen Verbreitung von Java-Programmen ist diese Option ein eher Kandidat für ein Modul oder kann ganz weggelassen werden.

### **Compile kernel as ELF - if your GCC is ELF-GCC**

dient zur Unterscheidung zwischen den verschiedenen Compilerversionen. Falls auf Ihrem System ein gcc-Compiler der Version 2.7.0 oder neuer installiert ist, sollte der Kernel im ELF-Format übersetzt werden. Bei einem älteren Compiler ist „n“ die richtige Antwort. Das Kommando `gcc -v` gibt die Versionsnummer des installierten Compilers aus.

### **Processor type**

gibt an, für welchen Prozessor der Kernel optimiert werden soll. Der Wert „386“ erzeugt einen Kernel, der auf allen Intel-Prozessoren der 80x86-Reihe funktioniert. Die Werte „Pentium“ und „PPro“ können nur bei gcc-Versionen 2.7.0 und neuer verwendet werden.

## **Floppy, IDE and other block devices**

### **Normal floppy disk support**

dient zur Auswahl, ob der Treiber für Diskettenlaufwerke in den Kernel compiliert werden soll. Angesichts der immer geringer werdenden Bedeutung dieses Mediums kann dieser Treiber auch gut als Modul gebaut werden. Das funktioniert natürlich nur, wenn der Kernel ausschließlich von der Festplatte gebootet werden soll.

### **Enhanced IDE/MFM/RLL disk/cdrom/tape support**

und

### **Old harddisk (MFM/RLL/IDE) driver**

lassen die Wahl zwischen dem neuen, schnelleren IDE-Treiber, der zudem Bandlaufwerke, CDROMs, PCMCIA-Festplatten und IDE-Wechselplatten unterstützt, und dem alten IDE-Treiber, der nur Festplatten verwalten kann. Falls keine Probleme auftauchen, sollte immer der neue Treiber verwendet werden.

### **Use old disk-only driver on primary interface**

ermöglicht die Verwendung des alten Treibers auf dem ersten Festplatten-Controller und des neuen Treibers auf allen weiteren im System installierten Controllern.

### **Include IDE/ATAPI CDROM support**

ist nur mit dem neuen IDE-Treiber verfügbar und ermöglicht die Verwendung von CDROMs am IDE-Bus.

### **Include IDE/ATAPI TAPE support**

ist ebenfalls nur mit dem neuen IDE-Treiber verfügbar und ermöglicht die Verwendung von Bandlaufwerken am IDE-Bus.

### **Support removable IDE interfaces (PCMCIA)**

In Verbindung mit dem PCMCIA-Paket von David Hinds können mit dieser Option PCMCIA-IDE-Festplatten verwendet werden. Das PCMCIA-Paket liegt auf [hyper.stanford.edu](http://hyper.stanford.edu) im Verzeichnis `/pub/pcmcia`.

### **CMD640 chipset bugfix/support**

Viele 486 und Pentium Rechner mit SiS oder Neptune Chipsatz verwenden diesen leicht fehlerhaften IDE-Controller. Im Zweifelsfall ist es besser, diese Option einzuschalten, um Abstürzen oder Datenverlusten vorzubeugen. Bei PCI-Systemen erkennt der Kernel automatisch einen CMD640 Chip, bei VESA-Bus Systemen muß dem Kernel der Bootparameter `ide0=cmd640_vlb` übergeben werden.

### **RZ1000 chipset bugfix/support**

Ebenso wie der CMD640 hat der RZ1000 Chip, der oft in Pentium-Rechnern mit Neptune-Chipsatz verwendet wird, einen groben Hardwarefehler, der zu Datenverlusten führen kann. Diese Option ermöglicht einen sicheren Betrieb dieses Kontrollers auf Kosten der Geschwindigkeit.

Die anderen IDE-Optionen dienen der Optimierung auf bestimmte Controllertypen oder ermöglichen das dritte oder vierte IDE-Interface anzusprechen. Sie erfordern zusätzlich Bootparameter.

### **Loopback device support**

läßt das „mounten“ einer Datei als Dateisystem zu. Dieses Feature erfordert ein *mount*-Programm der Version 2.5 oder neuer und hat nichts mit dem Netzwerk-„loopback-device“ zu tun.

### **Multiple devices driver support**

,

### **Linear (append) mode**

und

### **RAID-0 (striping) mode**

ermöglichen, mehrere Partitionen zu einer großen zu verbinden. *Linear mode* beschreibt die Partitionen nacheinander, d. h. wenn eine Partition voll ist, wird die nächste verwendet. *RAID-0 mode* beschreibt alle Partitionen parallel, was zu deutlichen Geschwindigkeitsvorteilen führt, wenn die Partitionen sich auf verschiedenen Festplatten befinden.

### **RAM disk support**

Eine Ramdisk funktioniert einen Teil des Arbeitsspeichers zur Festplatte um. Diese Funktion wird hauptsächlich bei der Installation verwendet, wenn auf der Festplatte noch kein Dateisystem zur Verfügung steht.

## **Initial RAM disk (initrd) support**

ermöglicht das Laden einer Ramdisk durch den Bootloader (lilo oder loadlin).

## **XT harddisk support**

bindet einen Treiber für steinalte Festplatten-Contoller ein, wie sie vor Jahren in den ersten IBM-PCs, auch als IBM-XT bekannt, verwendet wurden.

# **Networking options**

## **Network firewalls**

betreibt den Linux-Rechner als Firewall, wird normalerweise nicht benötigt.

## **Network aliasing**

ermöglicht die Angabe von mehreren IP-Adressen auf einem Netzwerkinterface, wird ebenfalls nur selten verwendet.

## **TCP/IP networking**

sollte fast immer eingeschaltet sein, da viele Programme, wie z. B. das X Window System, Netzwerkfunktionen verwenden, selbst wenn der Rechner an kein Netzwerk angeschlossen ist.

## **IP: forwarding/gatewaying**

versetzt den Rechner in die Lage, als „Brücke“ zu anderen Netzwerken für die Rechner des lokalen Netzes zu agieren. Diese Option wird ebenfalls benötigt, wenn der Rechner ein SLIP- oder PPP-Server für andere Rechner sein soll.

## **IP: PC/TCP compatibility mode**

kann eingeschaltet werden, falls Probleme mit Verbindungen zu DOS-Rechnern, die eine alte, fehlerhafte TCP-Software verwenden, auftreten.

## **IP: Allow large windows (not recommended if <16MB of memory)**

verbessert den Datendurchsatz auf sehr schnellen und langen Verbindungen, wie z. B. Satelliten-Verbindungen oder Überseestandleitungen, die mit Geschwindigkeiten von mehr als 2 MBit/s arbeiten. Diese Option erhöht die für Netzwerkpuffer verwendete Speichermenge drastisch und sollte auf Rechnern mit weniger als 16 MB RAM nicht eingeschaltet werden.

## **The IPX protocol**

muß verwendet werden, wenn der Rechner als Client oder Server in IPX (Novell-Netware) Netzen arbeiten soll. Zur Nutzung dieser Funktionalität sind noch andere Softwarepakete (mars\_nwe, oder lward) erforderlich, die die Konfiguration des IPX-Interfaces übernehmen. Zur Verwendung als Client muß das NCP-Dateisystem verwendet werden.

## **Appletalk DDP**

In Verbindung mit dem *netatalk* Software-Paket ermöglicht dieser Treiber, den Linux-Rechner als Datei- und Druckserver für Apple-Macintosh-Systeme zu verwenden. Ebenso können von Linux aus Drucker, die an einen Mac angeschlossen sind, verwendet werden. Linux kann jedoch nicht die Netzwerkdateisysteme der Macs als Client ansprechen.

Die restlichen Netzwerkoptionen dienen eher selten benötigten Spezialanwendungen und können im Normalfall ausgeschaltet werden.

# SCSI support

## **SCSI support**

bindet den Grundstock von Funktionen zur Steuerung von SCSI-Systemen (Small Computer Systems Interface) in den Kernel ein.

## **SCSI disk support**

bestimmt, ob SCSI-Festplatten unterstützt werden. Die Angaben zu den entsprechenden Controllern werden später erfragt.

## **SCSI tape support**

bindet die Unterstützung für SCSI-Bandlaufwerke in den Kernel ein.

## **SCSI CD-ROM support**

Hier kann die Unterstützung für SCSI-CD-ROM-Laufwerke eincompiliert werden. Diese Unterstützung ist nur für CD-ROMs vorhanden, die über einen der nachfolgend konfigurierten Hostadapter angesteuert werden.

## **SCSI generic support**

stellt ein allgemeines Interface zu SCSI-Geräten, die durch keinen der anderen speziellen Treiber angesprochen werden können, zur Verfügung. Diesem Treiber ordnet der Kernel alle SCSI-Geräte zu, die er nicht erkennt (z. SCSI-Drucker oder Scanner). In der Regel sind besondere Programme in der Benutzerebene zuständig, um das Gerät sinnvoll zu steuern.

## **Probe all LUNs on each SCSI device**

Eine LUN (Logical Unit Number) dient der Unterscheidung logischer Geräte innerhalb eines SCSI-Gerätes. Manche CD-Wechsler präsentieren jede der ansteuerbaren CDs als eigenes CD-ROM auf einer eigenen LUN. Die meisten normalen Geräte verwenden diese Möglichkeit jedoch nicht.

## **Verbose SCSI error reporting (kernel size +=12K)**

kann im Normalfall ausgeschaltet bleiben. Falls Probleme mit einem SCSI-Gerät oder Controller auftreten, kann diese Option verwendet werden, um leichter verständliche Fehlermeldungen vom Kernel zu bekommen.

# SCSI low-level drivers

Die Konfigurationsmöglichkeiten der SCSI-Hostadapter beziehen sich fast ausschließlich auf die herstellerabhängigen Varianten verschiedener, für PC-Systeme angebotenen Erweiterungskarten. Da die meisten Kartenhersteller noch keine linuxspezifischen Treiber ausliefern, hängt die Verwendbarkeit eines bestimmten Controllers von der direkten Unterstützung im Linuxkernel ab. Linux unterstützt über 30 verschiedene SCSI-Chipsätze auf über 100 Controllern unterschiedlichster Hersteller.

Da die Treiber für die unterschiedlichen SCSI-Controller fast durchweg einen recht hohen Speicherbedarf haben, sollte nur der Treiber fest in den Kernel eingebunden werden, der beim Booten benötigt wird. Alle anderen Treiber können als Module übersetzt und zur Laufzeit geladen werden.

Für die 53C8xx Chipsätze von NCR (üblicherweise als onboard-Controller auf vielen PCI-Systemen zu finden) stehen zwei Treiber zur Auswahl. Der erste wird mit `NCR53c7,8xx SCSI support` in den Kernel eingebunden und unterstützt zusätzlich zur 53C8xx-Familie auch noch die 53C7xx-Reihe von NCR. Dieser Treiber ist der von Drew Eckhardt entwickelte Linux-NCR-Treiber und wird nicht

mehr weiterentwickelt. Da dieser Treiber keine Paritätsprüfung auf dem SCSI-Bus zuläßt, sollte er nicht für FAST-SCSI-2 Anwendungen eingesetzt werden. Befindet sich in dem System lediglich ein Controller der 53C8xx Baureihe, sollte der zweite Treiber, der mit der Option `NCR53C8XX SCSI support` selektiert werden kann, eingesetzt werden. Er ist eine Portierung des FreeBSD-Treibers von Wolfgang Stanglmeier und Stefan Esser. Durch seine Fähigkeit zur Paritätsprüfung und „tagged command queueing“ ist er sowohl in FAST-SCSI-2 als auch in WIDE-SCSI Anwendungen verwendbar.

## **Network device support**

### **Network device support**

stellt die grundlegende Netzwerkfunktionalität sowie das Netzwerk-loopback-Interface, das Netzwerkverbindungen innerhalb des Rechners simuliert, zur Verfügung. Ist das System an kein Online-Netzwerk angeschlossen und wird auch keine Verbindung via SLIP, PPP oder PLIP zu einem anderen System aufgebaut, kann diese Frage verneint werden. Für Store-and-Forward Netzwerke wie UUCP, FIDO-NET als auch für TERM (einen IP-Emulator) braucht diese Option nicht installiert zu werden.

### **Dummy net driver support**

stellt ein zweites lokales Netzwerkinterface ähnlich dem loopback-Device zur Verfügung, das auf die eigene SLIP oder PPP IP-Adresse konfiguriert werden kann, um den Rechner auch bei nicht bestehender Netzwerkverbindung unter der eigenen IP-Adresse ansprechen zu können.

### **EQL (serial line load balancing) support**

ermöglicht, mehrere Netzwerkverbindungen parallel zum selben Ziel zu betreiben und das Datenaufkommen gleichmäßig auf alle Leitungen zu verteilen.

### **PLIP (parallel port) support**

ermöglicht, ein „mini“-IP-Netzwerk zwischen zwei Rechnern mit Hilfe der Druckerschnittstelle aufzubauen. Es wird ein besonderes Kabel für die Verbindung benötigt, das auch als „Turbo-Laplink“ Kabel bekannt ist. In der Datei *drivers/net/README?.plip* sind Schaltpläne für das Kabel zu finden. Die maximale Kabellänge beträgt 15m.

### **PPP (point-to-point) support**

bindet den PPP-Treiber für Wählverbindungen in den Kernel ein. PPP ist eine Weiterentwicklung des SLIP (Serial Line Internet Protocol). Falls der anzuwählende Rechner PPP unterstützt, ist diesem Protokoll der Vorzug zu geben.

### **SLIP (serial line) support**

ist eine Möglichkeit, sich via Modem über analoge Verbindungen in das Internet einzuwählen.

### **CSLIP compressed headers**

ermöglicht einen höheren Datendurchsatz und bessere Antwortzeit bei interaktiven Anwendungen, da die Verwaltungsinformationen in den Datenpaketen komprimiert werden. Diese Option darf nur verwendet werden, wenn die Gegenstelle ebenfalls Headerkompression unterstützt.

### **Keepalive and linefill**

ist eine Erweiterung zu SLIP, die zur Überwachung der Leitungsqualität insbesondere bei schlechten Analogverbindungen und Fernverbindungen dient. Die Gegenstelle muß dieses Verfahren unterstützen.

### **Six bit SLIP encapsulation**

ermöglicht die Verwendung von SLIP Verbindungen über serielle Leitungen, die nicht alle Zeichen korrekt übertragen, verringert jedoch den Datendurchsatz.

Die weiteren Optionen beziehen sich auf paketvermittelte Datennetze und Amateurfunkverbindungen. Weitere Informationen sind unter `Documentation/networking/` und `drivers/net/README.*` zu finden.

### **Ethernet (10 or 100Mbit)**

Die folgenden Optionen binden Treiber für verschiedene Ethernetkartentypen und Parallelportadapter in den Kernel ein. Viele „Noname“-Karten sind zu anderen Modellen (bei ISA-Karten meistens zu NE2000 und bei PCI-Karten meistens zu AMD/Lance) kompatibel.

### **Token Ring driver support**

bindet Unterstützung für Token Ring Netzwerke in den Kernel ein.

### **IBM Tropic chipset based adaptor support**

ist ein Treiber für Token Ring Hardware, basierend auf dem „Tropic“-Chipsatz von IBM. Unterstützung für andere Karten existiert derzeit nicht.

### **ARCnet support**

Im Gegensatz zu Ethernet Karten besitzen alle Arcnet Karten die gleiche Softwareschnittstelle, so daß mit diesem Treiber jede Arcnet Karte (von 320 Kbit/s bis 100 Mbit/s) verwendet werden kann.

### **Enable arc0e (ARCnet ``Ether-Encap'' packet format)**

ermöglicht die Kommunikation des Linux-Rechners mit Microsoft Lanmanager Rechner, die Arcnet verwenden.

### **Enable arc0s (ARCnet RFC1051 packet format)**

ermöglicht die Kommunikation des Linux-Rechners mit Rechnern, die das „alte“ Arcnet Protokoll verwenden, wie NetBSD (PC) und AmiTCP (Amiga).

## **ISDN subsystem**

### **ISDN support**

bindet ISDN (Integrated Digital Services Network) Unterstützung in den Kernel ein. Bislang werden nur drei ISDN-Kartentypen unterstützt, davon zwei aktive und eine passive Karte.

### **Support synchronous PPP**

erlaubt Verbindungen zu „Terminaladaptoren“ bzw. ISDN-Modems, die keine direkte Kapselung von IP-Paketen in HDLC können (ELSA, ZyXEL) und normalerweise über die serielle Schnittstelle oder den Druckeranschluß betrieben werden.

### **Use VJ-compression with synchronous PPP**

kann nur verwendet werden, wenn es von der Gegenstelle unterstützt wird und verringert die Antwortzeiten bei interaktiven Anwendungen.

### **Support generic MP (RFC 1717)**

ermöglicht das Versenden von verschiedenen Netzwerkpaketen (IPX, Appletalk) über das ISDN-Interface.

### **Support audio via ISDN**

Mit dieser Option kann der Rechner mit Hilfe zusätzlicher Software (vgetty, vbox) als digitaler



Anrufbeantworter oder Voice-Mailboxsystem betrieben werden.

### **ICN 2B and 4B support**

Ist ein Treiber für ICN ISDN-Karten. Die 2B Karte unterstützt 2 B-Kanäle, die 4B-Karte 4 B-Kanäle pro Karte. Die ICN Karten sind aktive Karten, die zusätzliche Firmware zum Betrieb benötigen und mit einem eigenen Programm geladen werden müssen.

### **PCBIT-D support**

Die PCBIT Karte ist wie die ICN Karten eine aktive Karte und benötigt zusätzliche Software. Sie unterstützt 2 B-Kanäle.

### **Teles/NICCY1016PC/Creatix support**

ist ein Treiber für alle Teles-8, Teles-16 und Teles-16.3 kompatiblen ISDN Karten. Zur Konfiguration ist weitere Software erforderlich. Der Treiber besitzt im Gegensatz zu den anderen beiden Karten in Deutschland keine Postzulassung.

## **CDROM drivers (not for SCSI or IDE/ATAPI drives)**

Hinter diesem Menü verbergen sich zahlreiche Treiber für CD-ROM-Laufwerke, die mit einer eigenen Erweiterungskarte betrieben oder an Soundkarten angeschlossen werden. Die meisten dieser Treiber erfordern zusätzliche Bootparameter, die die I/O-Adressen und Interruptnummern angeben, auf die das CD-ROM eingestellt wurde. IDE und SCSI CD-ROMs müssen hier nicht angegeben werden.

## **Filesystems**

### **Quota support**

ermöglicht das Setzen von Nutzungsgrenzwerten für den Festplattenplatz einzelner Benutzer. Dieses Feature ist nur für Systeme sinnvoll, die von vielen "speicherhungrigen" Personen benutzt oder als Dateiserver verwendet werden. Bislang werden Quotas nur auf Partitionen unterstützt, die mit dem Ext2 Dateisystem formatiert wurden.

### **Mandatory lock support**

wird von einigen Datenbanksystemen wie Oracle und Informix verwendet. Diese Option erfordert die neuesten Versionen der Programme mars\_nwe, nfsd, samba, netatalk und anderer Dateiserver-Programme. Sie wird im Normalfall nicht benötigt.

### **Minix fs support**

Dieses Dateisystem sollte wegen seiner weiten Verbreitung auf jeden Fall in den Kernel eincompiliert werden, da es meistens auf Disketten verwendet wird. Es ist in der Partitionsgröße auf 64 MB beschränkt.

### **Extended fs support**

Dieses Dateisystem wird nur noch aus Gründen der Kompatibilität unterstützt. Auf neu installierten Systemen sollte es nicht mehr verwendet werden.

### **Second extended fs support**

Das „Second extended Filesystem“ ist das schnellste, leistungsfähigste und derzeit am weitesten verbreitete Linux-Dateisystem. Es erlaubt Partitionsgrößen bis zu 2TB (2 Terrabyte = 2000 Gigabyte) und Dateigrößen bis zu 2GB. Das ext2fs ist von vornherein auf gute Erweiterbarkeit ausgelegt und ist „das“ Linux-Dateisystem schlechthin.

### **xiafs filesystem support**

Das xiafs ist stark an das Minix-Dateisystem angelehnt und älter als das ext2fs. Es unterstützt Partitionsgrößen bis zu 4 GB und Dateigrößen bis zu 64 MB. Dateinamen können bis zu 255 Zeichen lang sein. Das Dateisystem wird heute kaum noch verwendet und ist hauptsächlich aus Kompatibilitätsgründen noch verfügbar.

### **DOS FAT fs support**

liefert die Grundlage für alle anderen auf den von MS-DOS verwendeten „File Allocation Tables“ basierenden Dateisysteme.

### **MSDOS fs support**

Wenn MS-DOS-Disketten oder -Partitionen gemountet werden sollen, kann das MS-DOS-Dateisystem eincompiliert werden. Es enthält keine Unterstützung für komprimierte Dateisysteme. Wird diese Frage mit n beantwortet, ist es trotzdem möglich, MS-DOS-Disketten mit den *mtools* zu bearbeiten.

### **VFAT (Windows-95) fs support**

ist im Prinzip das gleiche wie das MS-DOS Dateisystem, nur mit Unterstützung für lange Dateinamen und Symlinks, wie sie unter Windows `95 verwendet werden. Es enthält keine Unterstützung für komprimierte Dateisysteme.

### **umsdos: Unix like fs on top of std MSDOS FAT fs**

Das umsdos (UNIX/MS-DOS) Dateisystem erzeugt mit Hilfe versteckter Dateien alle Funktionen, die ein UNIX Dateisystem benötigt. Es kann zur Installation von MS-DOS und Linux auf ein und derselben Partition verwendet werden, ist aber wesentlich langsamer als die anderen UNIX Dateisysteme.

### **/proc filesystem support**

Das sog. Prozeßdateisystem bildet die interne Prozeßstruktur des Kernels auf das Dateisystem ab. Die neuen Versionen von *ps*, *free*, *xsysinfo*, u. a. verwenden dieses Dateisystem, um die entsprechenden Informationen zu lesen.

### **NFS filesystem support**

Wird das NFS-Dateisystem mit eingebunden, so können im Netz Verzeichnisse anderer Rechner, die als NFS-Server betrieben werden, gemountet werden.

### **Root file system on NFS**

ermöglicht sog. „Diskless Workstations“, Rechner ohne Festplatte unter Linux zu betreiben.

### **BOOTP support**

ermöglicht das Bestimmen der IP-Adresse von Rechnern, die ohne eigene Festplatte betrieben werden und alle Dateisysteme via NFS einbinden. Um diese Funktion verwenden zu können, muß ein BOOTP-Server im Netzwerk zu Verfügung stehen. Falls das BOOTP-Protokoll im EPROM der Ethernetkarte verwendet wird und den Kernel mit allen notwendigen Informationen versieht, muß diese Option nicht installiert werden.

### **RARP support**

ist der Vorläufer des BOOTP-Protokolls.

### **SMB filesystem support (to mount WfW shares etc..)**

Mit dem SMB Dateisystem können exportierte Festplatten und Verzeichnisse von Rechnern in das Dateisystem eingebunden werden, die als LAN MANAGER Server dienen. Diese Server müssen allerdings LAN MANAGER auf TCP/IP-Basis unterstützen.

### **SMB long filename support (EXPERIMENTAL)**

stellt lange Dateinamen (bis zu 255 Zeichen) unter dem SMB-Dateisystem zur Verfügung. Dieses Feature funktioniert nicht mit Rechnern, die unter Windows 3.11 (Windows for Workgroups) arbeiten.

#### **NCP filesystem support (to mount NetWare volumes)**

versetzt den Linux-Rechner in die Lage, von Novell-Servern exportierte Verzeichnisse anzusprechen.

#### **ISO9660 cdrom filesystem support**

Das ISO9660 Dateisystem ist das Standarddateisystem für CD-ROMs. Linux unterstützt sowohl das 'reine' ISO9660 Dateisystem als auch die Rock Ridge Erweiterungen.

#### **OS/2 HPFS filesystem support (read only)**

Dieser Treiber ermöglicht das Lesen von OS/2 HPFS-Dateisystemen. Ein Beschreiben der Dateisysteme ist nicht möglich.

#### **System V and Coherent filesystem support**

ermöglicht den Zugriff auf SystemV R2 386, Xenix und Coherent Dateisysteme.

#### **Amiga FFS filesystem support (EXPERIMENTAL)**

wird auf dem PC wohl in den seltensten Fällen benötigt. Es bietet Zugriff auf Festplatten des Commodore AMIGA Systems. AMIGA Disketten können nicht gelesen werden, da der AMIGA ein anderes physikalisches Aufzeichnungsformat verwendet, das von PC-Floppy-Controllern nicht unterstützt wird.

#### **UFS filesystem support (read only)**

Das UFS-Dateisystem wird überwiegend von BSD-verwandten Systemen wie NextStep, SunOS oder FreeBSD verwendet. Auch einige System V Varianten können UFS-Partitionen lesen und schreiben. Dieser Treiber erlaubt nur Lesezugriff auf UFS-Partitionen.

#### **BSD disklabel (FreeBSD partition tables) support**

FreeBSD verwendet ein eigenes Partitionsschema für Festplatten. Mit dieser Erweiterung ist Linux in der Lage, FreeBSD Partitionstabellen zu lesen und in Verbindung mit dem UFS-Dateisystem auch Partitionen zu mounten.

#### **SMD disklabel (Sun partition tables) support**

Wie FreeBSD, verwendet auch SunOS ein eigenes Partitionsformat. Sollen SunOS Festplatten unter Linux gelesen werden, muß diese Option eingeschaltet sein. Zusätzlich ist das UFS-Dateisystem erforderlich.

## **Charakter devices**

#### **Standard/generic serial support**

ist der Standardtreiber für serielle Schnittstellen. Er deckt die Ports COM1--COM4, AST fourport, passive BOCCA, passive Accent und HUB6 Karten ab. Werden mehr als die ersten beiden seriellen Schnittstellen (COM1 und COM2) verwendet, sollte der Treiber nicht als Modul übersetzt werden, da beim Entfernen des Moduls auch die Konfigurationsinformationen für alle Schnittstellen gelöscht werden. Zur Konfiguration der Schnittstellen ist das Programm *setserial* erforderlich.

Die nächsten Optionen sind Treiber für einige aktive Schnittstellenkarten verschiedener Hersteller. Für einige ist zusätzliche Firmware für den Betrieb unter Linux erforderlich.

#### **Parallel printer support**

steuert den normalen Druckeranschluß an. Falls das BIOS des Rechners verschiedene Betriebsmodi für den Druckeranschluß unterstützt, sollte die Einstellung „normal“ oder „Standard“ gewählt werden. Die Einstellung des IRQ für den Druckeranschluß erfolgt mit dem Programm *tunelp*.

### **Mouse Support (not serial mice)**

Bietet Unterstützung für verschiedene Busmäuse und Trackballs in Laptops. Die meisten Busmäuse verhalten sich wie PS/2-Mäuse.

### **Support for user misc device modules**

erlaubt das Laden von Modulen für andere Eingabegeräte. Bislang sind allerdings so gut wie keine Module dieser Art verfügbar. Es bietet sich an, diese Abfrage zu verneinen.

### **QIC-02 tape support**

compiliert den Treiber für QIC-02 Bandlaufwerke in den Kernel ein.

### **Do you want runtime configuration for QIC-02**

Diese Option ermöglicht es, die Parameter des Bandlaufwerkes zur Laufzeit einzustellen. Sie benötigen dazu das Programm *qic02conf*. Ansonsten müssen Sie die Parameter Ihres Laufwerkes in der Datei `include/linux/tpqic02.h` eintragen.

### **Ftape (QIC-80/Travan) support**

bindet einen Treiber für sog. „Floppytape“-Streamer in den Kernel ein. Obwohl diese Laufwerke ursprünglich am Disketten-Controller betrieben wurden, gibt es inzwischen auch eigene Erweiterungskarten für diese Laufwerke. Unterstützt werden der Colorado FC-10 und FC-20, der Mountain MACH-2 sowie die meisten 2Mbit/s Controller. Zur Verwendung eines dieser Controller müssen in der Datei `drivers/char/ftape/Makefile` die entsprechenden Optionen aktiviert werden.

### **Advanced Power Management BIOS support**

ist fast ausschließlich für batteriebetriebene Laptops interessant, deren BIOS der APM-1.0 oder APM-1.1 Spezifikation entspricht. Die meisten BIOSe der „großen“ Rechner entsprechen nicht den APM Spezifikationen und verwenden ein für den Linuxkernel reserviertes Datensegment, was zu Abstürzen beim Booten des Rechners führen kann. Der Treiber kann auch keine sog. „Energiespar-Monitore“ abschalten.

### **Watchdog Timer Support**

Ein Watchdog (Wachhund) ist ein Gerät, daß den Rechner nach einem Absturz automatisch neu startet. Ein Hardware-Watchdog besteht aus einer Steckkarte, die einen Hardwarereset durchführt, wenn die auf der Karte enthaltenen Zeitgeber nicht in regelmäßigen Abständen zurückgesetzt werden. Der Kernel besitzt auch die Möglichkeit, eine Watchdog-Karte durch Software zu emulieren, allerdings kann die Emulation den Rechner nicht immer neu starten, wenn der Absturz die Funktion des Rechners zu stark beeinträchtigt hat.

### **Enhanced Real Time Clock Support**

stellt die Statusinformationen der PC-Uhr unter `/proc/rtc` zu Verfügung. Über das Device `/dev/rtc` kann die Uhr gestellt werden. Ebenso wird die Einstellung der „Weckzeit“ ermöglicht, die viele Laptops verwenden, um sich zu einer bestimmten Uhrzeit wieder einzuschalten. Dieser Treiber ist besonders auf Multiprozessorsystemen wichtig.

# Sound

## Sound card support

bindet Treiber für diverse Soundkarten ein. Die Treiber erwarten die Angabe der verwendeten DMA, IRQ und I/O Einstellungen.

# Kernel hacking

## Kernel profiling support

stellt unter `/proc/profile` Informationen zur Verfügung, mit denen bestimmt werden kann, wieviel Zeit der Kernel für bestimmte Funktionen benötigt.

## Profile shift count

bestimmt den Adressabstand, mit dem die einzelnen vom Kernel ausgeführten Befehle in `/proc/profile` aufgelistet werden.

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [Fremde Welten](#) **Up:** [Systemverwaltung](#) **Previous:** [Der Protokollschreiber syslogd](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

# Fremde Welten

Linux ist nur eines von vielen Betriebssystemen für PC, meistens ist es weder das erste noch das einzige auf der Festplatte. Mit dieser Tatsache wird in der Linux-Gemeinde souverän umgegangen: nur wer DOS kennt, weiß die Qualitäten von Linux wirklich zu schätzen.

Die Betriebssysteme sind eine Sache, Anwendungsprogramme eine ganz andere. Allen Mängeln von DOS und Windows zum Trotz ist das Angebot an Software für diese Systeme von einer beeindruckenden Vielfalt und Qualität.

Das Angebot an professionellen Anwendungen für Linux beschränkt sich zur Zeit noch überwiegend auf Entwicklungswerkzeuge. Wer nicht auf die Fertigstellung neuer Produkte für Linux warten will, hat verschiedene Möglichkeiten, Programme von anderen Betriebssystemen unter Linux einzusetzen. Wir stellen nun drei wichtige Ansätze vor: **dosemu**, **Wine** und die **iBCS2**-Emulation.

- 
- [dosemu](#)
    - [Allgemeines](#)
    - [Verlassen des Emulators](#)
  - [Wine](#)
  - [Der iBCS2-Emulator](#)
    - [Wie Sie iBCS2 bekommen und installieren können](#)
    - [Shared Libraries](#)
    - [Gerätedateien](#)
    - [Programme installieren](#)

**Next:** [Wine](#) **Up:** [Fremde Welten](#) **Previous:** [Fremde Welten](#)

## Subsections

- [Allgemeines](#)
    - [Warnung!](#)
    - [Funktion:](#)
    - [Syntax:](#)
    - [Optionen:](#)
    - [Diskettenlaufwerke](#)
    - [Das „virtuelle“ Bootlaufwerk](#)
    - [Festplattenlaufwerke](#)
    - [Video-Konfiguration](#)
    - [Tastatur-Konfiguration](#)
    - [Serielle Schnittstellen](#)
    - [Terminalunterstützung](#)
    - [X-Window-Unterstützung](#)
  - [Verlassen des Emulators](#)
- 

# dosemu

## Allgemeines

### Warnung!

**Dosemu** ist ALPHA-Software, d. h. er befindet sich noch in der Testphase seiner Entwicklung. Sichern Sie deshalb ihre Daten, **bevor** Sie ihn starten, und verwenden Sie ihn **keinesfalls**, um für Sie wichtige Daten **ohne Sicherungskopie** zu bearbeiten.

### Funktion:

**Dosemu** ist ein Programm, das auf Benutzerebene arbeitet und bestimmte Fähigkeiten des Linux-Kernels und des 80386-Prozessors ausnutzt, um MS-DOS innerhalb von Linux auszuführen.

Der Name „**dosemu**“ ist irreführend, da es sich nicht um einen MS-DOS-Emulator handelt, sondern um ein Programm, das die Hardwareumgebung nachbildet, in der MS-DOS normalerweise arbeitet. Es handelt sich eigentlich um einen „PC-Emulator“.

**Zum Betrieb des dosemu wird eine MS-DOS-Lizenz ab der Version 3.3 oder DR-DOS 6.0 benötigt.**

**Dosemu** kann, wenn er unter Superuser-Rechten auf der Linuxkonsole ausgeführt wird, direkt auf die Geräte Festplatte, Grafikkarte, Lautsprecher, Tastatur sowie ausdrücklich freigegebene IO-Ports zugreifen. Der direkte Hardwarezugriff beschleunigt den Ablauf der MS-DOS-Programme wesentlich, verringert aber die Sicherheit des Systems gegenüber Abstürzen und Datenverlusten. Wird **dosemu** unter „normalen“ Benutzerrechten ausgeführt, erfolgen alle Ein- und Ausgaben über Linux-Betriebssystemfunktionen. Dadurch sind weder Grafik noch der IBM-PC-Zeichensatz verfügbar.

Diese Beschreibung bezieht sich auf die **dosemu**-Version 0.53. Das Format der Konfigurationsdatei hat sich gegenüber der Version 0.49 sehr stark verändert, so daß alte Konfigurationsdateien nicht weiterverwendet werden können.

Syntax:

```
dos [-234ABCKNVcgkmst] [-DDebugOptionen] [-M Größe] [-P Datei] [-e Größe] [-x Größe] [2> Debugdatei]
```

Optionen:

- 2 286 CPU emulieren
- 3 386 CPU emulieren
- 4 486 CPU emulieren
- A von dem ersten Floppylaufwerk booten
- B von dem zweiten Floppylaufwerk booten
- C von der ersten Festplatte booten (normalerweise „ hdimage“)

-D Optionen

setzt die Debugmaske; „+“ schaltet die Ausgabe ein, „-“ schaltet sie aus (-D-a+dv schaltet alle Debugausgaben außer denen zu Plattenzugriffen und Bildschirmausgaben aus)

d	Festplatte	v	Bildschirmausgabe	R	Lesezugriffe
k	Tastatur	i	Portzugriffe	W	Schreibzugriffe
s	RS-232	p	Druckerport	h	Hardware
w	Warnungen	g	Allgemeines	x	XMS
m	Maus	I	IPC	E	EMS
c	Konfiguration	X	X-Window-Support	D	DPMI

\end{rawhtml} \item[-F \Argument{ Konfigurationsdatei}] zwingt {\bf dosemu,} seine Konfiguration aus der als {\it Konfigurationsdatei\} angegebenen Datei zu lesen. \item[-K] Tastaturinterrupt ({\sf int9}) freischalten\footnote{Einige residente Programme, wie z.\thinspace B.\ Sidekick, erwarten diesen Interrupt.} \item[-M \Argument{Größe}] setzt die Speichergröße auf ""{\it Größe\}"" Kilobyte \item[-N] Kein DOS booten; dient nur zur Fehlersuche \item[-P \Argument{Datei}] kopiert die Debugausgaben in ""{\it Datei\}"" \item[-T \Argument{ Verzeichnis}] setze Verzeichnis für temporäre Dateien \item[-V] Einschalten der BIOS{\bindestrich}VGA{\bindestrich}Emulation (schließt {\sf -c} ein) \item[-X] X{\bindestrich}Unterstützung; hat denselben Effekt, als würde {\bf dosemu} unter dem Namen ""{\bf xdos}"" aufgerufen \item[-Y \Argument{ Terminalfifo}] reserviert\footnote{Die Option wird von einem neuen Startprogramm zur X{\bindestrich}Unterstützung verwendet und bezeichnet einen FIFO zur Terminal- und Maussteuerung.} \addtocounter{footnote}{-1} \item[-Z \Argument{ Mausfifo}] reserviert\footnotemark \item[-c] schaltet die direkte Bild\schirm\aus\ga\be auf die Grafikkarte ein\footnote{ermöglicht den IBM{\bindestrich}Zeichensatz, sowie DOS{\bindestrich}Textmodus{\bindestrich}Farben; erfordert Superuser{\bindestrich}Rechte} \item[-d] gibt das aktuelle Terminal wieder frei und startet den {\bf dosemu} auf einer eigenen virtuellen Console. \item[-e \Argument{Größe}] stellt ""{\it Größe\}"" Kilobyte EMS zur Verfügung \item[-g] schaltet die Grafikerunterstützung ein; z.\thinspace B.\ für Spiele \item[-k] schaltet den direkten Tastaturzugriff ein \item[-m] schaltet die Mausunterstützung ein \item[-s] verwendet den neuen Code zur Unterstützung anderer Bildschirmgrößen als 80x25 (noch nicht vollständig) \item[-t] schaltet den Timer{\bindestrich}Interrupt ({\sf int8}) frei\footnote{Einige residente Programme wie z.\thinspace B.\ Bildschirmschoner verwenden diesen Interrupt.} \item[-v \Argument{ Grafikkartentyp}] Als Grafikkartentyp kann "1" für VGA{\bindestrich}, "2" für EGA{\bindestrich}, "3" für CGA{\bindestrich} und "4" für MDA{\bindestrich}Karten angegeben werden. \item[-x \Argument{Größe}] stellt ""{\it Größe\}"" Kilobyte XMS zur Verfügung \item[2> \Argument{Datei}] leitet alle Debugmeldungen in ""{\it Datei\}"" um; wird diese Option weggelassen, so lenkt {\bf dosemu} automatisch alle Debugmeldungen nach {\sf /dev/null} um \end{optlist} \subsection{ Voraussetzungen} \index{dosemu!Anforderungen} Um den {\bf dosemu} zu betreiben, muß Ihr System folgende Software{\bindestrich}Vo"-raus"-set"-zun"-gen erfüllen: \begin{enumerate}



\item einen Linux{\bindestrich}Kernel Version 1.1.12 oder neuer, mit  
 IPC{\bindestrich}Unterstützung\footnote{IPC{\bindestrich}Unterstützung kann als Option beim Übersetzen des  
 Kernels ausgewählt werden} \item den GNU{\bindestrich}C{\bindestrich}Compiler gcc 2.5.8 oder neuer \item die  
 Linux{\bindestrich}C{\bindestrich}Library ab der Version 4.5.21 \item MS{\bindestrich}DOS Version 3.3 oder  
 neuer (DR{\bindestrich}DOS 6.0 funktioniert ebenfalls) \end{enumerate} \subsection{Übersetzen des dosemu}  
 \index{dosemu!Compilieren} Wechseln Sie in das Verzeichnis, in dem sich die {\bf dosemu}{\bindestrich}Quelltexte  
 befinden. Falls Ihre Linux{\bindestrich}Installation nicht dem Filesystem{\bindestrich}Standard entspricht, müssen  
 Sie im \mbox{Makefile} die Pfade zu den Header{\bindestrich}Dateien und Libraries an Ihre Installation anpassen.  
 {\bf make} überprüft automatisch, ob die Header{\bindestrich}Dateien und Libraries für X vorhanden sind, und  
 bindet X{\bindestrich}Unterstützung in {\bf dosemu} ein. %Die einzige Option, die beim Compilieren des {\bf  
 dosemu} zu setzen ist, %ist die X{\bindestrich}Window{\bindestrich}Unterstützung. Wenn Sie nicht über eine  
 X{\bindestrich}Installation %verfügen, setzen Sie im Makefile die Variable {\tt X\_SUPPORT = 0} %(Zeile 12).  
 Standardeinstellung ist {\tt X\_SUPPORT = 1}. Mit dem Kommando {\sf make doeverything} wird {\bf dosemu}  
 compiliert und installiert. Falls auf Ihrem Rechner kein TeX installiert ist, übersetzen Sie {\bf dosemu} mit dem  
 Kommando {\sf make most}. {\bf make} versucht, {\bf dosemu} nach dem Übersetzen zu installieren; deshalb sollte  
 dieses Kommando mit Superuser{\bindestrich}Rechten ausgeführt werden. Nach dem Übersetzen kopieren Sie die  
 Datei {\sf ./examples/config.dist} nach {\sf etc} und benennen sie in {\sf dosemu.conf} um. \subsection{Die  
 Laufzeitkonfiguration} \index{dosemu!Konfiguration} Die Datei {\tt /etc/dosemu.conf} enthält die Konfiguration des  
 {\bf dosemu}. Sie besteht aus Konfigurationsanweisungen, die einen Parameter oder eine Liste aus Unteranweisungen  
 mit Parametern erwarten. Anweisungen mit einem Parameter haben die Syntax: \\centerline{ {\sf Anweisung}~~~{\it  
 Parameter}\} Anweisungen, die eine Liste erwarten, haben die Syntax:\\centerline{ {\sf Anweisung} {\it  
 Unteranweisung1} [{\it Parameter1}\]} Unteranweisung2 [{\it Parameter2}\]} ... \} } Kommentare beginnen mit einem  
 ""{\sf \#}"" und enden am Zeilenende. Bei den Anweisungen wird Groß{\bindestrich} und Kleinschreibung nicht  
 unterschieden. Bestimmte Argumente, wie z.\thinspace B.\ Da\~{-}tei\~{-}na\~{-}men und die Druckeroptionen, werden jedoch  
 unverändert übernommen und sind deshalb auf eine {\bf exakte} Schreibweise angewiesen. \subsubsection{Tastatur,  
 Ports und andere Kleinigkeiten} \index{dosemu!/etc/dosemu.conf@{\tt /etc/dosemu.conf}} \begin{optlist}  
 \item[dosbanner {\it on/off\}]~\\ Schaltet die Begrüßungsmeldung des {\bf dosemu} an oder aus. \item[ipxsupport  
 {\it on/off\}]~\\ Wenn Sie von {\bf dosemu} aus auf ein Novellnetz zugreifen wollen, können Sie mit diesem  
 Schalter die IPX/SPX{\bindestrich}Emulation einschalten. Innerhalb des {\bf dosemu} muß dann IPX.COM nicht  
 mehr geladen werden. LSL.COM oder IPXODI.COM werden nicht emuliert. Damit diese Option funktioniert, muß  
 IPX im Linuxkernel eingeschaltet sein. \item[pktdriver novell\\_hack]~\\ Ermöglicht eine Umsetzung von  
 Novell{\bindestrich}8137{\bindestrich}Paketen zu raw{\bindestrich}802.3{\bindestrich}Paketen.  
 \item[allowvideoportaccess {\it on/off\}]~\\ Gibt den Zugriff auf die IO{\bindestrich}Ports der Grafikkarte frei.  
 \item[bootA, bootC]~\\ Gibt an, von welchem Laufwerk der {\bf dosemu} MS{\bindestrich}DOS laden soll. Diese  
 Anweisungen erwarten keinen Parameter. Von ihnen darf nur {\bf eine} in der Konfigurationsdatei vorkommen.  
 \item[EmuSys {\it Endung\}], EmuBat {\it Endung\}]~\\ Gibt die Dateiendungen für die Dateien CONFIG.SYS und  
 AUTOEXEC.BAT für \mbox{{\bf dosemu}} an. So können sich auf demselben Laufwerk zwei verschiedene  
 Startupdateien für DOS befinden, je nachdem, ob MS{\bindestrich}DOS direkt oder im {\bf dosemu} gebootet wird.\\  
 Beispiel:\\ {\tt EmuSys EMU}\\ {\tt EmuBat EMU}\\ Wenn DOS im {\bf dosemu} gebootet wird, werden die Dateien  
 CONFIG.EMU und AUTOEXEC.EMU verwendet, ansonsten die normalen CONFIG.SYS und AUTOEXEC.BAT.  
 \item[cpu {\it Typ\}]~\\ Bestimmt, welcher Prozessor emuliert werden soll. Als ""{\it Typ\}"" kann ""{\tt 2}"" , ""{\tt  
 3}"" oder ""{\tt 4}"" angegeben werden. Es ist ebenfalls möglich, den ganzen Prozessornamen anzugeben (""{\tt  
 80286}"" , ""{\tt 80386}"" und ""{\tt 80486}""). \item[debug {\ {\sf Flag} {\it Wert\} {\sf Flag} {\it Wert\} ... \}]~\\  
 Als Unteranweisungen sind {\sf config, port, video, serial, printer, disk, read, write, keyb, mouse, warning, general,  
 xms, ems, dpmi, hardware} und {\sf IPC} zulässig, die jeweils als Parameter {\sf on} oder {\sf off} erwarten.\\  
 Beispiel: \\begin{rawhtml}

```

debug  {  config  on    disk   off    warning  off    hardware  on
          port    on    read   off    general  off    IPC        off
          video   on    write  off    xms      off
          serial  off   keyb   on    ems      off
          printer off   mouse  on    dpmi    off
        }

```

HogThreshold *Microsekunden*

Gibt die Anzahl Microsekunden an, die mindestens zwischen zwei Tastaturabfragen verstreichen müssen. **Dosemu**  
 benutzt diesen Zeitabstand, um eine Tastaturschleife zu erkennen. Ein hoher Wert spart Rechenzeit unter Linux und  
 verschlechtert die Performance des Emulators.

FastFloppy *Wert/off*

Der Wert bezeichnet die Zeit zwischen Aktualisierungen der Daten auf dem Diskettenlaufwerk in Sekunden. Um den optimierten Diskettenzugriff abzuschalten, muß als *Wert* „off“ angegeben werden.

mathco *on/off*

Über diese Option wird dem **dosemu** mitgeteilt, ob ein Coprozessor vorhanden sein soll. Diese Option hat nichts mit einem wirklich im Rechner installierten Coprozessor zu tun. Sie kann beliebig gesetzt werden und ist vor allem für MS-DOS-Programme interessant, die unbedingt einen Coprozessor erfordern, da Linux einen Coprozessor emuliert.

ports *Liste*

Erlaubt dem Emulator den Zugriff auf die in „*Liste*“ angegebenen I/O-Ports. „*Liste*“ ist eine von „{ }“ eingeschlossene, durch Leerzeichen getrennte Liste der freizugebenden I/O-Adressen. Die Adressen können entweder hexadezimal oder dezimal angegeben werden. Hexadezimale Adressen werden durch Voranstellen von „0x“ vor die Adresse gekennzeichnet.

Beispiel: ports { 0x21e 0x22e 0x23e 0x24e 0x25e 0x26e 0x27e 0x28e }

speaker *Modus*

Setzt die Betriebsart des Lautsprechers. Mögliche Modi sind: „off“ - kein Ton, „emulated“ - bei jedem Ton wird lediglich ein „BEL“ (7) gesendet, „native“ - gibt den Zugriff auf die Ports 0x42 und 0x61 frei und ermöglicht so den „gewohnten DOS-Sound“. Diese Einstellung sollte nur bei direktem Arbeiten auf der Konsole verwendet werden.

timint *on/off*

Legt fest, ob der Timerinterrupt (*int8*) weitergeleitet wird. Viele Programme benötigen diesen Interrupt.

umb\_max *on/off*

Die Option erhöht die Bereitschaft des **dosemu**, freie Bereiche in den oberen Speicherbereichen zu finden und zu aktivieren.

ems *Größe/off*

Stellt die angegebene Menge EMS-4.0-Speicher unter DOS zur Verfügung. Kann auf off gesetzt werden.

Neuerdings wird auch folgende Syntax unterstützt:

ems {ems\_size *Größe* ems\_frame *Adresse*}.

Hierbei steht *Größe* wiederum für die Größe des EMS-Speichers, die zur Verfügung gestellt werden soll, und *Adresse* für die Basisadresse der Speicherseite, auf der die EMS-Speicherseiten eingeblendet werden.

hardware\_ram *Liste*

Benennt **dosemu** Speicherbereiche zwischen 640 KB und 1 MB, in die RAM eingeblendet werden soll. Der Ausdruck *Liste* ist eine von „{ }“ eingeschlossene Liste von Adressen. Jede Adresse bezeichnet den Anfang einer 4 KB großen Speicherseite. Größere Bereiche können durch die Verwendung des Schlüsselwortes *range* angegeben werden.

Beispiel: hardware\_ram {0xc8000 range 0xcc000 0xcffff} blendet in die Speicherbereiche von Adresse 0xc8000 - 0xc8fff und 0xcc000 - 0xcffff RAM ein. In diese Bereiche können ab MS-DOS 5.0 mit den Kommandos „highdevice“ und „loadhi“ Treiber und Programme geladen werden.

xms *Größe/off*

Richtet entsprechend der „Extended Memory Specification 3.0“ die als „*Größe*“ angegebene Menge XMS ein. Kann auf off gesetzt werden.

dpmi *off/off*

Schaltet die DPMI-Unterstützung (DOS Protected Mode Interface) im **dosemu** ein bzw. aus. Die DPMI-Unterstützung ist allerdings noch nicht vollständig implementiert und verwendbar.

# Diskettenlaufwerke

**Dosemu** kennt zwei Arten von Laufwerken: Diskettenlaufwerke und Dateien, die als Diskettenlaufwerk angesprochen werden. **Dosemu** ordnet die unter DOS verwendeten Laufwerksbuchstaben den Laufwerken in der Reihenfolge ihrer Konfiguration zu.

Syntax:

```
„floppy { Schlüsselwort [Wert] [...] }“
```

Zulässige Schlüsselwörter sind:

device *Blockdevice*

Definiert das „*Blockdevice*“ als Diskettenlaufwerk, z. B. „/dev/fd0“.

file *Dateiname*

Gibt an, daß die Datei „*Dateiname*“ ein Bild einer Diskette enthält und als Diskettenlaufwerk konfiguriert werden soll. Kommt diese Konfiguration zum Einsatz, so sollte die Anzahl der Sektoren, Köpfe und Spuren des „originalen“ Laufwerks angegeben werden, auf dem die Vorlage für das Image erstellt wurde.

sectors *Anzahl*

Definiert die Anzahl der Sektoren auf einer Spur des Laufwerks.

heads *Anzahl*

Gibt die Anzahl der Schreib- und Leseköpfe eines Laufwerks an.

tracks *Anzahl*

Beschreibt die Anzahl der Spuren, die ein Laufwerk umfaßt.

threeinch

Das mit dieser Option bezeichnete Laufwerk wird im vom **dosemu** emulierten CMOS-RAM als 3,5-Zoll-Laufwerk angezeigt.

fiveinch

Das mit dieser Option bezeichnete Laufwerk wird im vom **dosemu** emulierten CMOS-RAM als 5,25-Zoll-Laufwerk angezeigt.

readonly

Erlaubt nur Lesezugriffe auf das Laufwerk.

Beispiele:

```
floppy { Heads 2 Sectors 18 Tracks 80 threeinch file /usr/dos/fimage }  
floppy { Device /dev/fd0 threeinch }
```

Durch diese Einträge wird die Datei `/usr/dos/fimage` als Laufwerk A: und das „echte“ erste Laufwerk des Rechners als Laufwerk B: konfiguriert. Im „CMOS-RAM“ des **dosemu** tauchen beide Laufwerke als 3,5-Zoll-Laufwerke auf.

## Das „virtuelle“ Bootlaufwerk

Wenn **dosemu** mit einem Diskimage gebootet werden soll, Sie aber unter DOS trotzdem alle Diskettenlaufwerke verwenden wollen, kann ein besonderes „Bootlaufwerk“ konfiguriert werden. Dieses Laufwerk kann dann mit den Programmen *booton.com* und *bootoff.com* im laufenden **dosemu** ein- und ausgeblendet werden. Die Syntax ist identisch mit der Konfiguration eines Diskimage als Diskettenlaufwerk:

Syntax:

```
„bootdisk { Schlüsselwort [Wert] [...] }“
```

Beispiel:  
bootdisk { heads 2 sectors 18 tracks 80 threeinch file /dos/bootdisk }

## Festplattenlaufwerke

**Dosemu** kennt vier verschiedene Möglichkeiten, um auf Festplatten zuzugreifen.

1.

### Zugriff auf die gesamte Festplatte

Diese Zugriffsart erlaubt **dosemu**, die gesamte Festplatte anzusprechen (incl. des „Master Boot Records“). **Dosemu** versucht automatisch, die Festplattenparameter herauszufinden. Schlägt der Versuch fehl, müssen die Parameter manuell, wie bei den Diskettenlaufwerken beschrieben, angegeben werden. Diese Konfiguration ist extrem unsicher und sollte nur zu experimentellen Zwecken verwendet werden.

2.

### Zugriff auf eine Partition der Festplatte

Diese Zugriffsart ist unter dem Gesichtspunkt der Implementierung ähnlich dem Zugriff auf die gesamte Festplatte und arbeitet ebenfalls über den DOS-Interrupt int13. Sie ist jedoch wesentlich sicherer, da **dosemu** so keinen Zugriff auf andere Partitionen erhält und auch den „Master Boot Record“ nicht ansprechen kann. Um diese Zugriffsart verwenden zu können, muß eine Datei namens `./var/lib/dosemu/partition` existieren, die die benötigten Zusatzinformationen für den **dosemu** enthält. Die Datei kann mit dem `„mkpartition“`-Kommando erstellt werden. Um z. B. die Datei für die erste Partition der ersten Festplatte zu erstellen, führen Sie das Kommando `„mkpartition /dev/hda 1“` aus.

3.

### Eine als Festplatte „getarnte“ Datei

Wie bei den Diskettenlaufwerken ist es auch möglich, eine Festplatte durch eine Datei im Linux-Dateisystem zu simulieren. Festplattenimages werden mit dem Kommando `„mkhddimage“` erstellt. Mit `„mkhddimage -c 300 -h 12 -s 17 > hddimage“` erstellen Sie beispielsweise eine ca. 29MB große „Festplatte“ auf Ihrem Linux-Dateisystem. Die Nachteile dieser Methode liegen darin, daß auf „Festplatten“ dieser Art nur mit **dosemu** zugegriffen werden kann, und daß die Datei `hddimage` nur wachsen, aber nicht kleiner werden kann.

4.

### Zugriff auf das Linux-Dateisystem mit `lredir.exe`

Das Programm blendet einen beliebigen Verzeichnisbaum des Linux-Dateisystems als Festplatte ein. Auf diese Weise kann z. B. die DOS-Partition unter Linux gemountet und dann mit Hilfe von `lredir.exe` als Festplatte dem **dosemu** zur Verfügung gestellt werden. So kann sogar ein gleichzeitiger Zugriff auf die DOS-Partition von Linux und **dosemu** aus ermöglicht werden. Das Einblenden eines Linux-Dateibaums als Festplatte C: erfolgt durch den Aufruf von `lredir` im laufenden **dosemu**, z. B. mit:

```
lredir c:\linux\fs\usr\dos,
```

wobei der Prefix „linux\fs,“ dem tatsächlichen Pfad im Linux-Dateisystem voranzustellen ist.

Die Syntax der Festplattenkonfiguration ist derjenigen der Diskettenlaufwerke ähnlich.

Syntax:

```
„disk { Schlüsselwort [Wert] [...] }“
```

Zulässige Schlüsselwörter sind:

`wholedisk` *Festplatte*

Konfiguriert den Zugriff auf eine direkt angesprochene Festplatte. Als Festplatte kann z. B. `./dev/hda` angegeben werden.

`partition` *Partition Partitionsnummer*

Erlaubt den direkten Zugriff auf eine Festplattenpartition. Der Parameter `„Partition“` bezeichnet ein Blockdepot, unter Linux z. B. `./dev/hda1`, und der Parameter `„Partitionsnummer“` die Nummer der Partition (1–4).

`image` *Dateiname*

Konfiguriert eine Datei als Festplatte. Sie muß zuvor mit `„mkhddimage“` erstellt werden.

`readonly`

Erlaubt nur Lesezugriffe auf das Laufwerk.

Ebenso können noch die Schlüsselwörter `Tracks`, `Sectors` und `Heads`, wie bei den Diskettenlaufwerken beschrieben, verwendet werden.

Beispiele:

```
„disk { partition /dev/hda1 1 readonly }``  
„disk { image /usr/dos/hdimage }``
```

Mit diesen Zeilen wird im **dosemu** die erste Partition der ersten Festplatte als Laufwerk C: konfiguriert. Laufwerk C: darf nur gelesen werden. Laufwerk D: wird durch die Image-Datei `„/usr/dos/hdimage``` emuliert.

## Video-Konfiguration

Syntax:

```
„video { Schlüsselwort [Wert] [...] }``
```

Erlaubte Schlüsselwörter sind:

`mda`, `cga`, `ega`, `vga`

Bezeichnet die Art der Grafikkarte. Es darf nur eines der Schlüsselwörter angegeben werden.

`chipset` *Art*

Bezeichnet den auf der Grafikkarte verwendeten Chipsatz. Im Moment werden nur `„et4000```, `„trident```, `„diamond``` und `„s3``` erkannt.

`memsize` *Größe*

Gibt die Größe des Bildschirmspeichers an.

`vbios_file` *Datei*

Benennt eine Datei, die eine Kopie des Video-BIOS enthält. Diese Option ist in erster Linie hilfreich, um mit verschiedenen BIOS-Typen zu experimentieren. Eine BIOS-Datei kann mit dem Kommando `„/usr/lib/dosemu/getrom > vbios``` erzeugt werden.

`vbios_copy`

Der **dosemu** kopiert das Video-BIOS in das Emulator-RAM. Er verwendet dazu die Datei `„/dev/mem```, d. h. er muß mit Superuser-Rechten ausgeführt werden.

`vbios_seg`

Falls das VGA-BIOS Ihres Rechners nicht bei `0xc000` liegt, kann mit dieser Option eine andere Basisadresse angegeben werden.

`graphics`

Erlaubt dem **dosemu**, in Grafikmodi zu schalten. Diese Option funktioniert bislang nur mit VGA-Karten. Sie erfordert ebenso, daß der Chipsatz korrekt angegeben wurde, da sonst evtl. erweiterte Textmodi von Super-VGA-Karten beim Wechseln der virtuellen Konsole oder beim Verlassen des **dosemu** nicht richtig wiederhergestellt werden.

`console`

Ermöglicht die direkte Bildschirmausgabe und Grafikdarstellungen. Diese Option wird ignoriert, wenn der **dosemu** nicht auf der Konsole läuft.

Beispiel:

```
video { vga console graphics chipset et4000 memsize 1024 }
```

# Tastatur-Konfiguration

Syntax:  
„keyboard { Schlüsselwort [*Wert*] [...] }“

Die folgenden Schlüsselwörter werden erkannt:  
layout *Tastaturlayout*

Als *Tastaturlayout* werden

finnish	de	sf	sg	uk
finnish-latin1	de-latin1	sf-latin1	sg-latin1	us
dk	fr	es	portuguese	be
dk-latin1	fr-latin1	es-latin1	dvorak	no

unterstützt, wobei „de“ und „de-latin1“ für das deutsche Tastaturlayout stehen.

keybint *on/off*

Schaltet Tastaturinterrupts ein. Einige Programme, wie z. B. sidekick, benötigen diese Einstellung.

rawkeyboard *on/off*

Erlaubt eine sehr „echte“ Emulation der DOS-Tastatur incl. Sondertasten, wie z. B. *ALT*, *ALT-GR*, *Rollen*, *Pause*, *Druck*.

Im rawkeyboard-Modus stehen einige besondere Tastenkombinationen zur Verfügung:

## Strg-Rollen

schreibt die ersten 0x32 Interruptvektoren in die Debugausgabe

## Alt-Rollen

zeigt die vm86-Register

## RechtsShift-Rollen

erzeugt einen *int8* (timer tick)

## LinksShift-Rollen

erzeugt einen *int9* (Tastatur)

## Strg-Alt-BildAuf

rebootet den **dosemu**

## Strg-Alt-BildAb

beendet den **dosemu**

## Pause

hält den Emulator an und startet ihn auch wieder

## Strg-Alt-Fn

schaltet zu einer anderen virtuellen Konsole

# Serielle Schnittstellen

Syntax:  
„serial { Schlüsselwort [*Wert*] [...] device *Schnittstelle* }“  
com *Nummer*

Bezeichnet die COM-Schnittstelle unter DOS, die emuliert werden soll, wobei *Nummer* 1, 2, 3 oder 4 sein kann entsprechend COM1 - COM4 unter DOS.

mouse

Wird diese Option einer Schnittstelle zugeordnet, so schließt **dosemu** die Schnittstelle bei einem Wechsel der

virtuellen Konsole, um die gleichzeitige Verwendung einer Maus unter **dosemu** und dem X-Window-System zuzulassen.

base *IO-Adresse*

Gibt die Basisadresse der Schnittstelle an, die **dosemu** emulieren soll. Die Basisadresse muß nicht mit der tatsächlichen Adresse der Schnittstelle übereinstimmen.

irq *Nummer*

Gibt den IRQ an, auf dem **dosemu** die Schnittstelle emuliert.

device *Schnittstelle*

Gibt die tatsächliche Schnittstelle an, auf die zugegriffen werden soll, z. B. /dev/cua0.

Beispiele:

```
serial { mouse com 1 /dev/cua0 }  
serial { com 2 irq 3 base 0x2F8 /dev/cua3 }
```

Die erste Zeile macht **dosemu** COM1 bekannt und schaltet den Maus-Modus ein. Die zweite Zeile bewirkt, daß **dosemu** die 4. serielle Schnittstelle des Rechners auf COM2 abbildet.

## Terminalunterstützung

**Dosemu** kann auch in einer telnet-Session oder über eine Modemleitung verwendet werden. Für diesen Fall kann die Ausgabe von **dosemu** an die Fähigkeiten des Terminals angepaßt werden.

Syntax:

```
terminal { Schlüsselwort [Wert] [...] }
```

charset *latin/ibm*

Wählt den Zeichensatz aus, auf den **dosemu** die Ausgaben der DOS-Programme abbildet.

color *off/xterm/normal*

Beschreibt die Farbfähigkeiten des Terminals: *off* für monochrome Terminals, *xterm* für Terminals, die nur acht Farben unterstützen, und *normal* für IBM-PC-Konsolen und Terminals mit 16 Farben (z. B. ansi\_xterm).

updatefreq *Wert*

*Wert* gibt das Intervall zwischen den Bildschirmupdates in 1/20 Sekunden an.

updateline *Wert*

Gibt die Anzahl Zeilen an, die auf einmal aufgebaut werden. Je größer *Wert* gewählt wird, desto schneller baut **dosemu** den Bildschirm auf. Auf langsamen Modemleitungen ist es ratsam, diesen Wert kleiner zu wählen, um die Antwortzeiten des Emulators zu verringern.

method *fast/ncurses*

Legt die Terminalsteuerung fest. Bei *fast* sendet **dosemu** direkt ANSI-Sequenzen an das Terminal. Die Option *ncurses* sollte nur verwendet werden, wenn das Terminal keine ANSI-Sequenzen verarbeiten kann.

corner *on/off*

Schaltet das Schreiben von Zeichen in die rechte untere Bildschirmecke ein und aus. Viele Terminals scrollen den Bildschirm, wenn Zeichen in die rechte untere Bildschirmecke ausgegeben werden.

Beispiel: terminal { charset ibm color on method fast }

# X-Window-Unterstützung

Wird **dosemu** unter dem Namen *xdos* oder mit *dos -X* aufgerufen, so versucht er, das Programm *ansi\_xterm* zu starten und seine Aus- und Eingabe darüber abzuwickeln. Findet er das Programm nicht, so sucht er nach *color\_xterm* und *xterm*.

Syntax:

```
X { Schlüsselwort [Wert] [...] }
```

updatefreq *Wert*

*Wert* gibt das Intervall zwischen den Bildschirmupdates in 1/20 Sekunden an.

updateline *Wert*

Gibt die Anzahl Zeilen an, die auf einmal aufgebaut werden. Je größer *Wert* gewählt wird, desto schneller baut **dosemu** den Bildschirm auf. Auf langsamen Modemleitungen ist es ratsam, diesen Wert kleiner zu wählen, um die Antwortzeiten des Emulators zu verringern.

display *Displayname*

Benennt das X-Display, das **dosemu** verwenden soll. Wird diese Option nicht gesetzt, so nimmt **dosemu** das in der Umgebungsvariable *DISPLAY* genannte Display.

title *Titel*

Setzt den Titel für das DOS-Fenster.

icon\_name *Name*

Setzt den Namen, mit dem das Icon von **dosemu** bezeichnet wird.

keycode

Erlaubt **dosemu** Zugriff auf die von XFree86 verwendeten Tastaturcodes.

blinkrate *Wert*

Ermöglicht einen blinkenden Cursor. *Wert* gibt die Blinkfrequenz an.

font *Name*

Setzt einen anderen Zeichensatz.

Beispiel:

```
X { updatefreq 1 title ,,DOS in a BOX`` icon_name ,,dosemu`` }
```

## Verlassen des Emulators

Um den **dosemu** zu verlassen, führen Sie das in der Distribution enthaltene Programm *exitemu.com* aus, oder schicken Sie von einer anderen virtuellen Konsole mit dem **kill**-Kommando ein Signal *SIGHUP* oder *SIGTERM* an den **dosemu**. Senden Sie ein *SIGKILL* nur im äußersten Notfall, da der **dosemu** dann nicht mehr die Bildschirmeinstellungen zurücksetzen und die Logfiles auf die Platte schreiben kann.

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [Wine](#) **Up:** [Fremde Welten](#) **Previous:** [Fremde Welten](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)



# Wine

Dem Vorbild von SUNs Wabi folgend haben sich einige Programmierer daran gemacht, unter dem Titel ``Wine" einen freien Windows-Emulator für Linux und Unix zu schreiben.

Mit Wine ist es möglich, einzelne Anwendungen für MS-Windows direkt im X Window System zu starten. Die Fenster der Anwendung werden vom X Server dargestellt und vom Window Manager verwaltet. Das Wine-Programm übernimmt die Umsetzung der MS-Windows-Aufrufe in entsprechende Funktionen des X Window Systems. Indem Wine auf Benutzerebene mit dem X Server zusammenarbeitet, werden weder Kerneländerungen noch spezielle Benutzerrechte benötigt. Wine kommt ohne jede Windows-Installation aus, kann aber die .dll-Libraries und Ressourcen von Windows benutzen, wenn es installiert ist.

Obwohl der aktuelle Stand der Entwicklung immer noch als Alpha-Teststadium bezeichnet werden muß, sind die sichtbaren Ergebnisse beeindruckend. Das notepad-Programm, ein einfacher Texteditor aus der MS-Windows-Installation, arbeitet, wenn auch mit kleinen Fehlern. Größere Programme, wie zum Beispiel die Textverarbeitung `write` oder das Malprogramm `pbrush`, können noch nicht benutzt werden, das Angebot an Spielen für Linux läßt sich jedoch schon jetzt durch eine Reihe von PD- und Shareware-Produkten für Windows bereichern.

Es werden kontinuierlich Fortschritte bei der Emulation weiterer Windows-Funktionen gemacht, so daß wahrscheinlich schon in einer der nächsten Auflagen des Linuxhandbuches ein ausführlicheres Kapitel über Wine enthalten sein wird.

Wenn Sie sich für den aktuellen Entwicklungsstand des Windows-Emulators interessieren oder sich selbst an der Entwicklung beteiligen wollen, finden Sie Informationen und die aktuellsten Dateien auf `tsx-11.mit.edu:/pub/linux/ALPHA/Wine/` und auf allen FTP-Servern, die dieses Verzeichnis spiegeln.

## Subsections

- [Wie Sie iBCS2 bekommen und installieren können](#)
  - [Shared Libraries](#)
    - [SVR3/COFF](#)
    - [SVR4/ELF](#)
  - [Gerätedateien](#)
  - [Programme installieren](#)
- 

# Der iBCS2-Emulator

Linux ist auf Quelltextebene zu 99% kompatibel zu seinem Vorbild, dem System V Unix. Leider bieten die meisten Hersteller professioneller Unix-Software ihre Produkte noch immer nicht für Linux an. Deshalb ist mit dem iBCS2-Emulator die Möglichkeit geschaffen worden, professionelle Software von anderen PC-Unixen direkt unter Linux anzuwenden.

Die Idee, ausführbare Programme zwischen den verschiedenen PC-Unix-Derivaten auszutauschen, ist nicht neu. Die Grundlage für die systemübergreifende Verwendbarkeit der Software bilden zwei standardisierte Binärformate (COFF für SVR3 und ELF für SVR4) sowie die "Intel Binary Compatibility Specification 2", kurz iBCS2.

Leider hat die iBCS2-Spezifikation einige Mängel, und die Hersteller von PC-Unixen sind verschiedene Wege gegangen, um diese Mängel für ihre eigene Betriebssystemversion zu beheben. Dadurch ist das Ziel einer vollständigen Binärkompatibilität knapp verfehlt worden. Indem der Linux-iBCS2-Emulator die eine oder andere Variante, die "personality", einiger iBCS2-Implementationen nachvollzieht, können Binärdateien mehrerer Unix-Versionen unter Linux eingesetzt werden. Namentlich sind das SCO, System V Release 4, einfache System V Release 3 Programme sowie die speziellen Versionen für ISC, Wyse V/386 und Xenix V/386. Mit kleinen Veränderungen am Kernel können auch Binärdateien der freien BSD-Unixe genutzt werden. Das Hauptgewicht der Entwicklung liegt auf Wyse, SCO und SVR4.

## Wie Sie iBCS2 bekommen und installieren können

Die Binärkompatibilität zu System V Unix muß durch Kernelfunktionen hergestellt werden. Die zum Laden der Binärdateien im COFF- und ELF-Format verantwortlichen Funktionen sind in jedem Linux-Kernel enthalten. Der Rest, der eigentliche iBCS2-Emulator, wird als Laufzeitmodul in den Kernel eingefügt.

Das Modul muß für jede Kernelversion passend erzeugt werden. Damit das funktioniert, müssen Sie die Sourcen zu Ihrem Kernel installiert haben ☒ und die Quelltexte vom iBCS2-Modul compilieren. Nur für einige der älteren Kernelversionen (1.0 und frühere als 1.1.15) sind Patches am Kernelcode und die neue Übersetzung des Kernels notwendig.

Die aktuellen Sourcen vom iBCS2-Emulator finden Sie auf [tsx-11.mit.edu](http://tsx-11.mit.edu/pub/linux/ALPHA/ibcs2) im Verzeichnis `/pub/linux/ALPHA/ibcs2` und auf jedem FTP-Server, der dieses Verzeichnis spiegelt.

Sie können die Sourcen an jedem beliebigen Ort auspacken. Wenn Sie das bereits existierende Verzeichnis `/usr/src/linux/ibcs2` benutzen, wird das Modul bei jeder Kernelübersetzung automatisch miterzeugt.

Das iBCS2-Laufzeitmodul wird manuell erzeugt, indem in dem gerade ausgepackten Verzeichnis `./ibcs2/` das `make`-Kommando aufgerufen wird.

Das fertige Modul heißt `iBCS` und wird mit dem [insmod-Kommando](#) in den laufenden Kernel eingebunden.

Wie bei jedem Quelltextpaket finden Sie die wichtigen Informationen zum aktuellen Paket im README. Zusätzlich gibt es HINTS zur Installation von Software für den Emulator.

## Shared Libraries

Dynamisch gelinkte Programme sind keine Erfindung der Linux-Entwickler. Auch unter System V Release 3 und 4 gibt es Shared Libraries. Natürlich sind die Bibliotheken für Linux und die Unix-Varianten unterschiedlich. Wenn Sie dynamisch gelinkte Programme verwenden wollen, müssen Sie die passenden Bibliotheken bereitstellen.

### SVR3/COFF

Weil die Schnittstelle zwischen der Bibliothek und dem Kernel für SVR3 in der iBCS2 definiert ist, können Sie beispielsweise die Shared Libraries von SCO im Verzeichnis `/shlib` unter Linux installieren, wenn Ihre Lizenz das erlaubt. Ein Satz freier Bibliotheken für SCO, die auf der freien GNU Library basieren, wird hoffentlich bald fertiggestellt.

Zwei Libraries, die `libnsl` und die `libX11_s`, wird es nicht für Linux geben. In der Bibliothek `libnsl` sind spezielle Netzwerkfunktionen enthalten, die mit den von Linux (noch) nicht unterstützten Streams arbeiten. Glücklicherweise benutzen die SCO-Programme auf Sockets basierende Gerädateien für die Netzwerkzugriffe und können deshalb auch ohne diese Bibliothek unter Linux eingesetzt werden.

### SVR4/ELF

Für SVR4 sind freie Shared Libraries erhältlich. Sie finden die Sourcen und fertig übersetzte Pakete dort, wo es den iBCS2-Emulator gibt. Die Bibliotheken werden normalerweise im Verzeichnis `/usr/lib` oder in `/lib` installiert. Sie können dem dynamischen Linker `ld.so.1` einen anderen Suchpfad in der Umgebungsvariablen `LD_LIBRARY_PATH` angeben.

Bezüglich der Netzfunktionen gelten die gleichen Beschränkungen wie für SVR3. Dynamisch gelinkte Programme, die das X Window System benutzen, benötigen Shared Libraries, die Sie aus den entsprechenden XFree86-Distributionen für die SVR4-Unixen erhalten.

# Gerätedateien

Der iBCS2-Emulator stellt den Anwendungen zwei Gerätetreiber zur Verfügung, mit denen bestimmte Funktionen angesprochen werden können. Es sind Bestrebungen im Gange, die Allokierung der Hauptgerätenummern für Laufzeitmodule dynamisch zu machen. Zur Zeit müssen Sie die Gerätedateien noch mit den hier angegebenen Gerätenummern erzeugen.

```
# mknod /dev/socksys c 30 0
# ln -s /dev/socksys /dev/nfsd
# ln -s /dev/null /dev/X0R
# mknod /dev/spx c 30 1
# _
```

Die Netzwerkfunktionen von SVR3, die über die Dateien `/dev/socksys` und `/dev/nfsd` angesprochen werden, können vom iBCS2-Emulator bedient werden, obwohl Linux Streams nicht unterstützt.

Für Verbindungen zum X-Server erwarten die SVR3-Programme die Dateien `/dev/X0R` (das Zeichen zwischen X und R ist eine Null) und `/dev/spx`. Der iBCS2-Emulator umgeht die eigentliche Funktionalität dieser Gerätetreiber, indem er eine feste Verbindung von `/dev/spx` zum lokalen X-Server herstellt.

## Programme installieren

Auch wenn die Binärdateien prinzipiell unter Linux ausführbar sind, kann es bei der Installation eines kommerziellen Programmes, beispielsweise für SCO Unix, zu Schwierigkeiten kommen. Zur Zeit lassen sich nur solche Programme problemlos unter Linux installieren, die ohne spezielles Installationsprogramm auskommen oder mit eigenen Installationsprogrammen geliefert werden. Unter Linux gibt es noch kein mit `custom` vergleichbares Programm.

Die Installationsdisketten der kommerziellen Programme sind häufig ohne ein Dateisystem einfach mit `tar` beschrieben. Wenn das GNU-`tar` das Format nicht erkennt, versuchen Sie es mit dem Schalter `-i`, damit `tar` die leere erste Spur der Diskette ignoriert.

Als Installationsprogramme werden sehr häufig Shellscripts eingesetzt. Es gibt kleine Unterschiede zwischen der `bash` und der `sh` von System V, die zum Abbruch der Installation führen können. Mit der Kommandozeilenoption `-n` können Sie die `bash` veranlassen, ein Shellscript auf Fehler zu testen, ohne es auszuführen.

Wenn Sie das Programm fertig installiert haben, kann es zu weiteren Hindernissen kommen, weil die Programme für PC-Unix die Linux-Console nicht unterstützen. Sie können die Linux-Console auf den PC-Zeichensatz umschalten, indem Sie die Steuersequenz ``ESC- (U'` eingeben. Zum Linux-Zeichensatz zurück kommen Sie mit ``ESC- (B'`.

Wenn bei Ihrem Programm Terminfo-Dateien mitgeliefert werden, können Sie versuchen, durch Belegung der Umgebungsvariablen `TERM` mit `sco386`, `at386`, `vt100` oder ähnlichen Einstellungen eine befriedigende Bildschirmdarstellung zu erreichen.

Die Linux-Version 1.2 unterstützt weiterhin maximal eine doppelte Belegung der Funktionstasten. Es gibt aber bereits Kernelpatches, die diesen Mangel beheben. Die Belegung der Funktionstasten mit

Funktionssequenzen weicht in jedem Fall von der bei SCO benutzten ab. Wenn Sie keine Möglichkeit haben, die entsprechenden Einstellungen für Ihr Programm zu verändern, können Sie auch mit dem Kommando `loadkeys` eine neue Tastaturtabelle mit geänderten Funktionssequenzen in den Kernel laden.

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [Datenreisen und reisende Daten](#) **Up:** [Fremde Welten](#) **Previous:** [Wine](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

# Datenreisen und reisende Daten

Vernetzung, das ist ein großes Thema in der computerisierten Gesellschaft unserer Zeit. Arbeitsplatzrechner werden zu „Local Area Networks“ (LAN) zusammengeschlossen, um mit einem gemeinsamen Datenbestand zu arbeiten und um den Informationsfluß in einem Unternehmen oder in einer Organisation schneller und effektiver zu machen. Sogenannte „Wide Area Networks“ (WAN) verbinden Rechner und LANs über Gebäudegrenzen hinweg. Im Internet, dem Netz der Netze, sind einige zigtausend Rechner weltweit 24 Stunden am Tag miteinander verbunden, um Daten und Dienste auszutauschen und sich Arbeit zu teilen.


Linux ist ein Kind des Internets. Ohne dieses weltweite Computernetzwerk hätte es sich niemals so schnell entwickeln können. Wenn es überhaupt entstanden wäre, würde kaum jemand außerhalb des Fachbereiches Informatik an der Universität von Helsinki davon wissen...

In den News, den öffentlichen Nachrichten des USENET, wurde Linux vorgestellt und wird seitdem ununterbrochen diskutiert. Hier werden Fragen von allgemeinem Interesse beantwortet, neue Programme und Versionen angekündigt und Projekte organisiert und koordiniert.

Das Betriebssystem ist längst nicht mehr das Produkt eines einzelnen großartigen Programmierers. Viele fähige Leute weltweit steuern ihren Teil zum Gelingen des großen Projekts bei. Wenige haben sich jemals gesprochen, geschweige denn gesehen. Die private Kommunikation findet mit elektronischer Post - der eMail - statt. Außerdem gibt es offene automatische Postverteiler, sogenannte „mailing lists“, mit denen Diskussionen in kleinen Gruppen möglich sind.

Die Verteilung neuer Versionen des Betriebssystems und der Programme geschieht ebenfalls über das Netz. Auf vielen Rechnern, vor allem an den Universitäten, wird ein umfangreicher Bestand an Linux-Software öffentlich angeboten. Diese Daten können mit dem `ftp`-Tool (File Transfer Protocol) auf jeden Rechner im Internet kopiert werden. Einige Sites bieten ihre Daten über Mailserver an, mit denen es auch den Teilnehmern eines reinen UUCP-Netzwerkes möglich ist, darauf zuzugreifen. Zur Entlastung der internationalen Datenleitungen werden die Linux-Verzeichnisse einiger zentraler ausländischer Server auf vielen Rechnern in der Bundesrepublik automatisch - teilweise täglich - „gespiegelt“.

Bei diesen beeindruckenden Möglichkeiten ist der Wunsch naheliegend, an diesem Netzwerk teilzunehmen. Linux hat alle Voraussetzungen für diese Teilnahme von UNIX „geerbt“. In den UNIX-Systemen ist die Vernetzung ein uraltes Thema. Datenfernübertragung und Kommunikation, wie sie auf DOSen mit „Mailboxen“ und ähnlichen Programmen nur auf Kosten der gesamten übrigen Funktionalität möglich ist, ist bei Linux einfach ein Teil des Betriebssystems. Als

Multisuser-Multitasking-System verwaltet Linux die serielle Schnittstelle ohne spürbaren Aufwand. 

Das „Protokoll“, auf dem die Vernetzung der großen UNIX-Rechner heutzutage beruht, heißt TCP/IP (Transmission Control Protocol/Internet Protocol). Es erlaubt direkte Verbindungen zwischen allen

an das Internet angeschlossenen Rechnern. Über diese Verbindungen wird (elektronische) Post direkt vom Absender zum Zielrechner geschickt, mit ihnen können Dateien zwischen Rechnern kopiert werden, die tausende Kilometer voneinander entfernt stehen, es können auch einfach interaktive Kommandos auf dem entfernten Rechner ausgeführt und die Ergebnisse lokal ausgegeben werden, indem die Parameter und Inhalte eines Fensters zum lokalen X11-Server übertragen werden. Im Linux-Kernel ist TCP/IP mit Netzwerktreibern für Ethernet, Druckerport und serielle Schnittstellen implementiert. Weitere Treiber, beispielsweise für ISDN-Karten, sind bereits so weit entwickelt, daß sie möglicherweise beim Erscheinen des Buches bereits im Kernel enthalten sind.

Einen eigenen Rechner oder ein lokales Ethernet „Connected To The Internet“ zu betreiben, ist allerdings sehr kostspielig. Der Preis für einen Internet-Anschluß erreicht leicht einen vierstelligen Betrag. Außerdem fällt einiger Verwaltungsaufwand an, der in einem öffentlichen Netz sehr ernst genommen werden sollte.

Der volle Luxus einer eigenen TCP/IP-Verbindung, gar einer Standleitung, in das *Wide Area Network* ist zur Befriedigung der realen Bedürfnisse zum Glück in der Regel nicht notwendig. Selbst in Firmen und Organisationen, die ein *Local Area Network* auf Ethernet-Basis betreiben, ist die direkte Anbindung an das Internet vor dem Hintergrund günstiger Alternativen meistens nicht lohnend.

In diesem Kapitel werden Werkzeuge und Methoden vorgestellt, die die Nutzung aller Internet-Services und die Einbindung des eigenen Rechners bzw. LANs in ein weltweites Computernetzwerk zu wesentlich günstigeren Konditionen erlauben.

Eine wachsende Zahl von Rechnerbetreibern bietet für Privatleute, Firmen und Organisationen Benutzeraccounts an, mit denen die volle Internetfunktionalität auf diesen Rechnern genutzt werden kann. Die Kosten für so einen Account betragen nur einen Bruchteil einer eigenen Internetverbindung und der Verwaltungsaufwand fällt komplett weg. Linux bietet die Möglichkeit, mit sogenannten Terminalprogrammen interaktiv auf solchen Rechnern zu arbeiten. Ergebnisse und Dateien können mit Dateitransferprotokollen vom Internetrechner auf den lokalen Rechner kopiert werden.

Zwei interessante Netzwerkdienste - eMail und News - können über ein älteres typisches UNIX-Netzwerk, das UUCP-USENET, mit einem minimalen Kosten- und Verwaltungsaufwand bezogen werden. Die dazu notwendigen Programme des UUCP-Pakets von Ian Taylor sind in allen größeren Linux-Distributionen enthalten. Das Netz privater UUCP-Rechner in der Bundesrepublik ist bereits heute sehr feinmaschig. Weil es sich bei den UUCP-Verbindungen nicht um Standleitungen handelt, haben prinzipiell alle so vernetzten Rechner die Möglichkeit, ohne zusätzlichen technischen Aufwand mehrere Rechner anzuschließen. Deshalb sind die Chancen, einen UUCP-Rechner im Nahbereich zu finden, der bereit ist, als Einwahlpunkt zu dienen, ziemlich groß.

- 
- [Technische Voraussetzungen](#)
    - [Modems](#)
    - [Serielle Schnittstellen](#)
  - [Kontaktaufnahme](#)
    - [Die minicom Terminalemulation](#)
    - [term](#)
  - [Seriell einloggen](#)

- [Das richtige getty](#)
- [getty\\_ps](#)
- [UUCP - Das Internet der Armen Leute](#)
  - [UUCP-Anschluß - aber wie?](#)
  - [Was kann UUCP?](#)
  - [Wie sicher ist UUCP?](#)
  - [Taylor-UUCP](#)
  - [Überblick über die Konfigurationsdateien](#)
  - [Log-Dateien](#)
  - [Die config-Datei](#)
  - [Die sys-Datei](#)
  - [Die port-Datei](#)
  - [Die dial-Datei](#)
  - [Testen der Konfiguration](#)
  - [Regelmäßige Verbindungen](#)
- [Elektronische Post mit \*smail\*](#)
  - [Wie sieht eine Mail denn nun aus?](#)
  - [Adressen, Adressen, Adressen](#)
  - [Taler, Taler, Du mußt wandern](#)
  - [Email-Software unter Linux](#)
  - [Installation von \*smail\*](#)
  - [Elektronische Post mit \*elm\*](#)
  - [Ein Test](#)
- [Usenet News](#)
  - [Die technischen Details](#)
  - [News-Software](#)
- [INN](#)
  - [INN und IP-Networking](#)
  - [Administrativer Kleinkram](#)
  - [Globale Parameter](#)
  - [Die Dateien \*active\* und \*newsgroups\*](#)
  - [Wer bekommt was - \*newsfeeds\*](#)
  - [Leben und Sterben des \*innd\*](#)
  - [Ein ausgehender Newsfeed über UUCP](#)
  - [Löschen von Artikeln mit \*expire\*](#)



- [Control-Messages](#)
- [Overview-Dateien](#)
- [Und jetzt ohne Hände...](#)
- [Der Newsreader \*tin\*](#)
- [Das Point-to-Point-Protokoll \(PPP\)](#)
  - [Voraussetzungen](#)
  - [Die Konfiguration des pppd](#)
  - [Die Verbindung starten](#)
  - [Die Verbindung beenden](#)

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [Technische Voraussetzungen](#) **Up:** [Das Linux Anwenderhandbuch](#) **Previous:** [Der iBCS2-Emulator](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

**Next:** [Kontaktaufnahme](#) **Up:** [Datenreisen und reisende Daten](#) **Previous:** [Datenreisen und reisende Daten](#)


## Subsections

- [Modems](#)
    - [Die technischen Daten](#)
    - [Kabelsalat](#)
  - [Serielle Schnittstellen](#)
    - [16550 UART](#)
    - [Mehrportkarten und Karten auf anderen Adressen/Interrupts](#)
- 

# Technische Voraussetzungen


Grundlage aller Vernetzung ist die physikalische Verbindung zweier Rechner. Das kann eine einfache Dreidraht-Nullmodem-Kopplung zweier serieller Schnittstellen sein, anstelle des Nullmodemkabels kann eine Telefonleitung mit zwei Modems stehen; es kann auch eine Ethernetverbindung sein, für die spezielle Hardware in den Rechner eingebaut werden muß.

## Modems

Für das Ziel - die Anbindung eines Rechners oder eines lokalen Netzes an ein „Wide Area Network“ - kommt eigentlich nur die Vernetzung über das bestehende Telefonnetz in Betracht. Dabei gibt es noch die Wahl zwischen der üblichen analogen Telefonverbindung und den wesentlich schnelleren digitalen Verbindungen auf Basis von ISDN. 

Um eine analoge Telefonleitung zur Datenfernübertragung (DFÜ) zu nutzen, bedarf es einer Einrichtung zur Umwandlung der logischen Daten und Zeichen aus dem Arbeitsspeicher des Rechners (Bits und Bytes) in Töne - und wieder zurück von Tönen in Bits und Bytes. Zu diesem Zweck gibt es sogenannte Modems (MODulator-DEModulator), die auf mehr oder weniger aufwendige Weise mehr oder weniger viele Daten transportieren können.


Früher (eigentlich gar nicht so lange her) wurde „gekoppelt“. Die von einem sogenannten Akustikkoppler erzeugten Töne wurden durch einen Lautsprecher in die Sprechmuschel des Telefonapparates gekrächzt, ein Mikrofon auf der anderen Seite nahm den Rest der übertragenen Geräusche auf, und der Demodulator destillierte daraus wieder Daten. Heute werden Akustikkoppler nur noch als Raritäten gehandelt. Ein echtes CCC-Modem Marke „Datenklo“ ist für einige Leute so wertvoll wie ein Hochgeschwindigkeitsmodem - wer sowas noch hat, verkauft es nicht.

Diesen legendären Werkzeugen der Pioniere folgten „echte“ Modems, die direkt an das Telefonnetz angeschlossen werden. Bis zum 1.7.1990 mußte jeder „Eingriff“ in das Fernmeldenetz von der P\*st selbst durchgeführt werden. Seither dürfen Modems mit BZT- oder FTZ-Nummer auch vom Benutzer selbst an eine TAE-Dose angeschlossen werden. 

# Die technischen Daten

Wer heute ein Modem kaufen möchte, findet beim lokalen Hardwarehändler oder im Anzeigenteil der aktuellen Ausgabe irgendeiner Computerzeitschrift jede Menge Angebote. Dabei tauchen Fachbegriffe und Zahlen auf, deren Bedeutung sich eigentlich erst bei der Benutzung, also nach dem Kauf, erschließt. Weil diese Begriffe zwangsläufig auch in diesem Kapitel benötigt werden, werden die wichtigsten erklärt:

## Baud / bps / cps

Diese Maßzahlen bezeichnen die Datenübertragungsrate, also die Geschwindigkeit, mit der die Information über die serielle Leitung geschickt wird. Die Begriffe *Baud* und *bps* werden häufig synonym verwendet, genau genommen bezeichnen sie aber unterschiedliche Parameter. Während die alte Einheit *Baud* die Anzahl der Wechsel zwischen Mark und Space und damit so etwas wie die physikalische Frequenz der Datenimpulse bezeichnet, bezieht sich *bps* auf den Durchsatz logischer Datenbits.  Die Maßzahl *cps* (characters per second) bezeichnet die tatsächliche Geschwindigkeit der Zeichenübertragung. Das Verhältnis von *bps* zu *cps* hängt von den Leitungsparametern ab und beträgt meist 10:1.

## Zeichenlänge, Parität, Stopbits

Beim PC werden jeweils 8 Stellen für die Repräsentation eines Bytes verwendet, es gibt aber durchaus andere Architekturen, die mit nur 7 Bit/Zeichen arbeiten. Speziell für die Übertragung von Texten zwischen Datenverarbeitungsmaschinen kann die Anzahl der Datenbits von 5 bis 8 Bit/Zeichen eingestellt werden. Um Binärdaten ohne zusätzliche Behandlung zwischen zwei PCs übertragen zu können, müssen 8 Bit/Zeichen eingestellt werden. Bei der Verbindung von UNIX-Rechnern im internationalen UUCP-Netz werden gelegentlich nur 7 Bit/Zeichen als Standard verwendet.

Ein Paritätsbit kann zur Erkennung von Übertragungsfehlern benutzt werden, indem die Quersumme der übertragenen Binärzeichen immer gerade bzw. ungerade ergänzt wird. Im Zeitalter moderner Modems mit automatischer Fehlerkorrektur hat diese Methode an Bedeutung verloren.

Das Ende jedes übertragenen Zeichens muß durch eine besondere Pause markiert werden. Die Länge dieser Pause wird durch die Stopbits eingestellt. Die hohe Auflösung der modernen Schnittstellenbausteine garantiert auch bei nur einem Stopbit immer eine zuverlässige Erkennung der Bytegrenzen.

## 8N1 / 7E1

Mit diesen Abkürzungen werden die beiden am häufigsten verwendeten Kombinationen für die Einstellung der Leitungsparameter bezeichnet. 8N1 bedeutet 8 Datenbits, keine Parität und 1 Stopbit, 7E1 bedeutet 7 Datenbits, gerade Parität und ein Stopbit.

## MNP


(Microcom Networking Protocol) In der MNP Protokollfamilie werden Verfahren zur Fehlerkorrektur (MNP2 bis MNP4) und zur Datenkompression (MNP5) festgelegt. Durch die automatische Fehlerkorrektur werden Übertragungsfehler bereits im Modem erkannt und die fehlerhaften Daten sofort neu angefordert. Durch die MNP5 Datenkompression werden die vom Computer gesendeten Bytes im Modem komprimiert, bevor sie durch das Nadelöhr - die Telefonleitung - geschickt werden. Auf der Gegenseite werden sie automatisch im Modem dekomprimiert. Mit MNP5 kann der Datendurchsatz unter günstigen Umständen verdoppelt werden. Dateien, die bereits mit `compress` oder `gzip` komprimiert sind, können von MNP5

nicht weiter gepackt werden. Falls es bei einem konkreten Modem sogar zu einer Verschlechterung der Übertragungsrate beim (MNP5-) Transfer komprimierter Dateien kommen sollte, können Sie für diesen Zweck die Datenkompression auch abschalten. Die dazu nötigen Befehle finden Sie in Ihrem Modemhandbuch.

## V42bis

Das V42bis ist ein Kompressionsverfahren wie MNP5. Die Kompressionsrate unkomprimierter Daten beträgt hier maximal 4.

## FAX

Die bekannten „Telefonkopierer“ oder Faxgeräte benutzen im Prinzip die gleiche Modulation-Demodulation, wie sie in Modems zur Datenfernübertragung benutzt wird. Allein die Daten und das „Protokoll“ sind unterschiedlich.  Die geeignete Software zur Bearbeitung der FAX-Bilder und zur Steuerung des Modems vorausgesetzt, lassen sich viele moderne Modems auch zum Senden/Empfangen von FAXen verwenden. Unter Linux bietet das mgetty/sendfax-Paket von Gert Döring diese Funktionalität.

## Hayes

Wie so häufig hat sich im Bereich der Modems ein herstellerspezifischer Befehlssatz zur Gerätesteuerung als Quasi-Standard etabliert. Im Bereich der Modems hat die Firma Hayes Microcomputer Products die Vorreiterrolle übernommen. Die meisten heute auf dem Markt befindlichen Modems sind *Hayes*-kompatibel, d.h. sie können mit den gleichen Befehlen programmiert werden wie die originalen Hayes-Modems. Die Hayes-Befehle sind leicht an dem typischen AT (für attention) als Einleitung für alle Befehle zu erkennen. Besonders wichtig sind die Befehle ATZ zum Reset des Modems und ATDP1234567 zum (Puls-)Wählen der Telefonnummer 1234567. Eine vollständige Erklärung aller Modemkommandos finden Sie im Gerätehandbuch des Herstellers.

## Handshake

Als Handshaking werden Verfahren zur Datenflußkontrolle bezeichnet. Bei der Verbindung Rechner-Modem-Modem-Rechner kann es leicht zu einer Situation kommen, wo ein Glied der Kette die Daten nicht so schnell verarbeiten kann, wie sie geschickt werden. Besonders am „Nadelöhr“ Telefonleitung kann es leicht zu so einem Datenstau kommen. Um das Überlaufen des Empfangspuffers und damit den Datenverlust zu verhindern, gibt es zwei gängige Methoden zur Datenflußkontrolle. Beim XON/XOFF Softwarehandshaking werden bestimmte Zeichen gesendet, um den Transfer von Zeichen zu drosseln (0x13 (^S) zum Anhalten und 0x11 (^Q) zum Starten). Diese Methode ist nur zum Übertragen von reinen Texten, zum Beispiel von ASCII-Terminals oder zu Druckern, geeignet, die diese Steuerzeichen nicht „aus Versehen“ enthalten.

Zur Verwendung mit Modems zur Datenfernübertragung ist das RTS/CTS Hardware-Handshake besser geeignet. Hierbei werden zwei Steuerleitungen des RS232-Kabels (Pins 5 und 6) benutzt, um der Gegenseite die Bereitschaft zum Senden bzw. zum Empfangen von Daten zu signalisieren. Wenn der PC Daten zum Modem senden will und bereit ist, Daten vom Modem zu empfangen, wird die RTS-Leitung HIGH gesetzt. Das Modem antwortet, indem es seine Bereitschaft, Daten vom Computer auf die Telefonleitung zu schicken, durch Hochsetzen der CTS-Leitung signalisiert. Wenn der Sendepuffer vom Modem vollgelaufen ist, setzt es die CTS-Leitung wieder LOW und veranlaßt den PC dadurch, den Datenfluß zu unterbrechen. Wenn umgekehrt der Computer nicht in der Lage ist, weitere Daten zu empfangen, setzt er die RTS-Leitung wieder LOW und veranlaßt dadurch das Modem, den Datenfluß anzuhalten.

## Carrier

Wenn eine Datenverbindung zwischen zwei Modems aufgebaut wird, senden beide Modems je einen Trägerton - den sogenannten Carrier - auf dem sie die zu übertragenden Daten aufmodulieren. Auch wenn keine Daten gesendet werden, kann ein Modem durch den Empfang des unmodulierten Carriers eine bestehende Verbindung erkennen. Solange dieser Ton auf der Telefonleitung empfangen wird, setzt das Modem die „Carrier Detect“-Leitung des RS232 Kabels auf HIGH und signalisiert dem Computer so eine bestehende Verbindung.

## Break

Ein Break ist ein spezielles Signal (keine Bitsequenz), mit der eine Ausnahmeroutine (Interrupt) bei der empfangenden Stelle ausgelöst werden kann. Der `getty`-Dämon benutzt dieses Signal zum Umschalten zwischen verschiedenen Leitungsgeschwindigkeiten beim Identifizieren einer ankommenden Modemverbindung.

## CR/NL

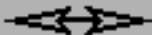
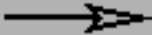
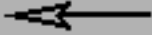
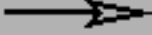

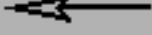




Der bei der Eingabe von Befehlen und Texten selbstverständlich notwendige Zeilenabschluß durch die `RETURN`-Taste wird nicht von allen seriellen Geräten gleich behandelt. Die in diesem Zusammenhang verwendeten Begriffe haben ihren Ursprung in der Zeit, als für die Datenausgabe noch Drucker und elektrische Schreibmaschinen verwendet wurden. Um eine neue Zeile zu beginnen, müssen bei diesen Geräten zwei Schritte ausgeführt werden: die Papiertransportwalze muß das Papier eine Zeile verschieben und der Druckkopf (Wagen) muß an den linken Rand zurückgefahren werden. Im ASCII-Standard sind Steuerzeichen für beide Bewegungen definiert. Der Wagenrücklauf (Carriage Return) wird von ASCII 0x0D (^M) ausgelöst, der Zeilenvorschub (Line Feed) von ASCII 0x0A (^J).

Weil eigentlich immer beide Bewegungen zusammen ausgeführt werden, denkt sich Linux automatisch zu einem Line Feed oder `NEWLINE` einen Wagenrücklauf dazu. Damit es bei der Kommunikation mit anderen Systemen und Geräten keine Probleme gibt, können ankommende und abgehende Zeilenenden in der einen oder anderen Weise übersetzt werden.

## Kabelsalat

Vernetzung hat immer auch irgendetwas mit Verdrahtung zu tun. Beim Modem sind gleich zwei Anschlüsse erforderlich: erstens muß das Modem an den Rechner angeschlossen werden, zweitens braucht das Modem eine Verbindung zum öffentlichen Telefonnetz.

Im RS-232-Standard wird die Verbindung einer „Endeinrichtung“ (Data Terminal Equipment, DTE) und einer „Kommunikationseinrichtung“ (Data Communication Equipment, DCE) vereinbart. Die serielle Schnittstelle des PC ist als DTE geschaltet, das Modem als DCE. Die folgende Tabelle zeigt die Belegung einer 25-poligen RS-232-Verbindung und zusätzlich die Belegung der 9-poligen Verbindung, wie sie meistens für die Maus verwendet wird.


Sub-D 25	Sub-D 9	Funktion	Symbol	DTE Richtung DCE
1	-	Schutzerdung	FG	
2	3	Sendedaten	TD	
3	2	Empfangsdaten	RD	
4	7	Request to Send	RTS	
5	8	Clear to Send	CTS	
6	6	Modem betriebsbereit	DSR	
7	5	Signalerdung	SG	
8	1	Carrier Detect	DCD	
20	4	Computer betriebsbereit	DTR	
22	9	RING Indikator	RI	

**Abbildung:** Die Steckerbelegung der RS-232 Schnittstelle

Das Modemkabel sollte alle aufgeführten Leitungen parallel wie ein Verlängerungskabel führen. Wie aus der Tabelle ersichtlich ist, werden für die eigentliche Datenübertragung nur zwei Leitungen verwendet (TD und RD), alle anderen Leitungen haben Steuerfunktionen. Es sind Kabel auf dem Markt, die nicht alle Steuerleitungen korrekt bedienen. Wenn die RTS/CTS-Leitungen für das Hardware-Handshaking eingespart werden, ist mit ernststen Problemen beim Betrieb von schnellen Modems zu rechnen.

Um zwei Rechner miteinander zu verbinden (DTE<-> DTE), müssen verschiedene Anschlüsse gekreuzt werden. Zu diesem Zweck sind sogenannte Nullmodemkabel erhältlich.

Bei den meisten Modems gehört ein Kabel zur Verbindung mit dem Telefonnetz (TAE-Dose) zum Lieferumfang. In der Regel haben die Modems zwei Western-Buchsen, eine für die ankommende Telefonleitung, eine zweite zum Anschluß eines Fernsprengerätes. Der zweite Anschluß wird automatisch getrennt, sobald das Modem „abhebt“. Bei den TAE Dosen gibt es unterschiedliche Buchsen, sogenannte *N*- und *F*-Kodierung. *F* steht dabei für Fernsprengerät, also den Telefonapparat, *N* für Nicht-Fernsprengerät, beispielsweise ein Modem. Um die in den TAE-Dosen mit *NF*-, *NFN*- oder *NFF*-Mehrfachbuchsen vorgesehene Serienschaltung mehrerer Geräte nutzen zu können, muß die durchgeschleifte Telefonleitung von der zweiten Westernbuchse zurück auf die dritte und vierte Leitung des ankommenden Kabels gelegt werden. Zu diesem Zweck gibt es im Fachhandel spezielle Kabel zu kaufen.

Wenn keine TAE-Dose mit Mehrfachbuchse vorhanden ist, kann in der Regel das mitgelieferte Anschlußkabel nicht verwendet werden, weil die Steckerkodierung nicht stimmt. In diesem Fall kann man die Anschlußdose von der Post durch eine passende (mit mehreren Buchsen) ersetzen lassen  oder einen Adapter bzw. ein neues Kabel kaufen. Ganz ignorante Menschen schnitzen einfach die Barten des Kodierschlüssels vom Stecker. Funktionieren tut's, ist aber nicht Sinn der Sache.

Unangenehm kann es auch werden, wenn ein nachträglich gekauftes TAE/Western-Adapterkabel falsch belegt ist. Die Firma Siemens beschäftigt hauseigene Philosophen zur Definition solcher Steckerbelegungen. Die Deutsche Telekom (TM) hat diese Steckerbelegung von ihrem ehemaligen Monopollieferanten übernommen und in irgendwelche Verordnungen geschrieben. Bei zugelassenen




Modems, insbesondere deutscher Hersteller, besteht also die realistische Möglichkeit, auf ein solches Exemplar zu stoßen.

## Serielle Schnittstellen

Der Standard-PC ist mit einer Multi-IO-Karte ausgerüstet, auf der manchmal der Festplatten- und Floppycontroller, meistens ein Drucker- und ein Gameport sowie zwei serielle Schnittstellen untergebracht sind.

Diese Karten sind zum Betrieb einer Maus immer ausreichend. Unter MS-DOS treten auch bei Modems selten Probleme auf. Unter Linux oder anderen Multitasking-Betriebssystemen (und auch bei MS-Windows) kommt es beim Betrieb von Hochgeschwindigkeitsmodems ab 9600 Baud zu vermehrten Übertragungsfehlern. Diese Fehler resultieren nicht aus einer schlechten Telefonleitung, sondern aus der verzögerten Bedienung der Schnittstelle durch das Betriebssystem.


Der Gerätetreiber für die seriellen Schnittstellen besteht aus zwei Ebenen. Auf der unteren Ebene werden die von der seriellen Leitung angebotenen Daten entgegengenommen. Im Unterschied zur Druckerschnittstelle, die in der Regel im „polling“ Betrieb arbeitet, also in einer Warteschleife den Druckerstatus aktiv so lange immer wieder abfragt, bis dieser Empfangsbereit ist, arbeitet die serielle Schnittstelle immer Interruptgesteuert. Der Schnittstellenbaustein löst einen Hardwareinterrupt aus, sobald ein Zeichen von der Schnittstelle empfangen wurde. Das bedeutet, daß der Programmablauf auf der unteren Ebene nicht mit dem Programmtext „synchron“, sondern durch äußere Ereignisse, die Hardwareinterrupts, „asynchron“ gesteuert wird. Obwohl die Interruptroutinen ohne


Kontextwechsel  sehr schnell ausgeführt werden, kann es zu kleinen Verzögerungen durch die Bearbeitung anderer Hardwareinterrupts kommen. Bei den schnellen Modems kann in dieser minimalen Zeitspanne bereits ein weiteres Zeichen eingetroffen sein.

Die einfachen Schnittstellenbausteine der billigen Multi-IO-Karten haben keinen eigenen Puffer, in dem sie dieses zusätzliche Zeichen speichern könnten. Deshalb wird einfach das bereits empfangene, aber noch nicht bearbeitete Zeichen überschrieben. Die Folge ist natürlich ein Datenverlust, der bei fehlerkorrigierenden Dateiübertragungen - etwa mit Z-Modem - erkannt wird. Die MNP-Fehlerkorrektur zwischen zwei Modems hat in diesem Zusammenhang keine Wirkung, weil der Verlust bei der Verbindung zwischen Modem und Computer auftritt.


Auf den meisten IO-Karten wird der Schnittstellenbaustein *16450 UART* oder *8250 UART* verwendet. Bei einer voll bestückten Karte (2 serielle Ports) sind diese Chips allein durch ihre Größe identifizierbar; es handelt sich um die beiden 40-poligen Teile. Die aufgedruckten Typenbezeichnungen müssen die oben genannten Zahlen enthalten. Auf den Multi-IO-Karten mit Floppy- und HD-Controller sind die beiden seriellen Schnittstellen meistens in einem Custom-Chip untergebracht. Bei allen Karten, die ab Werk nur mit einer seriellen Schnittstelle ausgeliefert werden, die aber mit einer zweiten Schnittstelle aufgerüstet werden können, ist ein Sockel für die Aufnahme des 16450 UART vorhanden.

### 16550 UART

Zu dem 16450/8250 UART Chip existiert ein pinkompatibler Bruder, der *16550AFN UART* von National Semiconductor.  Dieser Chip hat einem 16 Byte großen FIFO-Puffer.

Der *16550 UART* wird von Linux automatisch erkannt und unterstützt. Korrekt betrieben schreibt dieser Chip alle empfangenen Zeichen in seinen Puffer (First-In-First-Out). Der Interrupt wird erst ausgelöst, wenn der Puffer halb voll ist.  Das entlastet das Betriebssystem erheblich, weil nicht jedes Zeichen einzeln gelesen werden muß. Dadurch bleibt dem Kernel vor dem Überlaufen des Puffers in jedem Fall genügend Zeit, alle bereits wartenden Hardwareinterrupts höherer Priorität durchzuführen.

Speziell für den Betrieb von Hochgeschwindigkeitsmodems gibt es IO-Karten mit dem 16550 UART zu kaufen. Bei vielen internen Modems, die zwangsläufig eine serielle Schnittstelle mitbringen, wird auch der 16550 benutzt. Bei den oben erwähnten IO-Karten mit einem gesockelten 16450 kann dieser Chip einfach ausgetauscht werden (auf die richtige Polung achten). Das Auslöten eines 16450 kommt nur für erfahrene Bastler in Frage. Bei den Multi-IO-Karten mit Custom-Chip hilft gar nichts. Hier müssen die seriellen Schnittstellen abgeschaltet und durch eine Extrakarte ersetzt werden.

Die gleichzeitige Benutzung von vier seriellen Ports, wie sie durch die Bereitstellung von COM1 bis COM4 bei MS-DOS suggeriert wird, ist nicht möglich, weil sich jeweils COM1 und COM3 bzw. COM2 und COM4 einen Interrupt teilen. 

Die maximale Übertragungsrate für die serielle Schnittstelle liegt bei 115200 bps. Diese Rate kann nur nach einer entsprechenden Einstellung mit dem `setserial`-Programm eingestellt werden, indem die 38400 bps Rate ersetzt wird.

## Mehrportkarten und Karten auf anderen Adressen/Interrupts

Um mehr als zwei serielle Geräte betreiben zu können, unterstützt Linux spezielle Mehrportkarten, die durch geeignete Schaltung mehrere Schnittstellen mit nur einem Hardware-Interrupt betreiben, und solche Karten, die auf anderen Interrupts/IO-Ports arbeiten. Ohne Anspruch auf Vollständigkeit sind das AST FourPort und Accent Async Boards sowie die BOCA 4- und 8-Port Karten.

Seit Linux-Version 0.99.10 werden beim Systemstart nur die Standardports COM1 bis COM4 initialisiert, weil durch die Autokonfiguration an anderen Adressen Konflikte mit anderen, wichtigeren Karten auftreten können. Stattdessen müssen die zusätzlichen Schnittstellen mit dem `setserial`-Kommando initialisiert werden. Die Treiber sind im Kernel enthalten, können aber erst nach der Initialisierung benutzt werden.

Die Mehrportkarten arbeiten mit den gleichen Schnittstellenbausteinen wie die oben beschriebenen Multi-IO-Karten. Alles in diesem Zusammenhang über den 16450/8250 bzw. den 16550A Gesagte gilt hier ebenfalls.

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [Kontaktaufnahme](#) **Up:** [Datenreisen und reisende Daten](#) **Previous:** [Datenreisen und reisende Daten](#)



## Subsections

- [Die minicom Terminalemulation](#)
    - [Installation und Konfiguration](#)
    - [Anwählen eines anderen Rechners](#)
    - [Automatische Frage- und Antwort-Spiele](#)
    - [Dateien übertragen](#)
  - [term](#)
    - [Kurzer Überblick](#)
    - [Der term Server](#)
    - [Leitungstransparenz](#)
    - [trsh](#)
    - [tupload](#)
    - [tredir](#)
    - [txconn](#)
    - [tmon](#)
- 

# Kontaktaufnahme

Um die Netzwerkdienste wie eMail, News, telnet, ftp, irc und andere benutzen zu können, braucht man nicht gleich seinen eigenen Rechner zu vernetzen. Es reicht für den Anfang immer aus, als regulärer Benutzer auf einem anderen vernetzten Rechner eingetragen zu sein, der diese Dienste anbietet. Dafür muß man sich nicht körperlich zu dem anderen Rechner begeben, eine Modemverbindung dorthin reicht völlig aus.

Rechner, die solche Accounts anbieten, findet man in Mailboxlisten, wie sie in manchen Computermagazinen abgedruckt sind. Die in einigen der folgenden Beispiele aufgeführten Telefonlisten können ebenfalls als Einstieg dienen.

Für eine Datenreise „per Anhalter“ benutzt man Programme, die ein einfaches, seriell Terminal emulieren (nachahmen).


Bereits im frühen Unix wurden die zentralen Rechner über externe Geräte bedient. Zuerst waren das elektrische Schreibmaschinen (Teletypes oder *tty*'s), später Bildschirmterminals. Diese Geräte werden seit jeher über eine serielle Schnittstelle mit dem Zentralrechner verbunden.

Noch heute gibt es solche ASCII-Terminals, mit denen sehr günstig aus einem Linux-PC ein System mit mehreren Arbeitsplätzen gemacht werden kann. Die Arbeitsweise der Terminals ist primitiv. Im Prinzip schicken sie jedes auf der Tastatur eingegebene Zeichen unverändert an den Rechner und stellen alle vom Rechner geschickten Zeichen auf dem Bildschirm dar. Die meisten Terminals, z. B. das *vt100* von DEC, verstehen bestimmte Sonderzeichen, die sie als Befehle interpretieren und damit z. B. den Cursor positionieren, den Bildschirm löschen, zwischen Einfüge- und Überschreibmodus umschalten und sonst noch allerlei nützliche Sachen machen können.

Die Aufgabe eines „dummen“ Terminals kann auch von einem Computer übernommen werden. Dazu muß ein Programm gestartet werden, das die Funktionen eines Terminals nachbildet. Außer den Funktionen eines seriellen Terminals bieten die meisten Terminalemulationen noch eine Reihe weiterer Features wie Telefonlisten, Scriptprogramme und Dateitransfer, mit denen den erweiterten Möglichkeiten/Anforderungen dieser Nutzung entsprochen wird.

# Die minicom Terminalemulation

Das Programm minicom von Miquel van Smoorenburg ist eine Terminalemulation für den Linux-ASCII-Bildschirm. In der Bedienung ist sie dem DOS-Sharewareprogramm TELIX vergleichbar.

Die Installation und Konfiguration von minicom ist Sache der Systemverwalterin. Das fertig angepaßte Programm wird weitgehend interaktiv durch Tastaturkommandos bedient. Alle minicom Befehle werden durch CONTROL-A (^A) eingeleitet , den Hilfsbildschirm erhält man mit dem Kommando ^A-Z.

```
+=====+
|                                     |
|                               Minicom Command Summary                       |
|                                     |
|       Commands can be called by CTRL-A <key>                             |
|                                     |
|       Main Functions                Other Functions                        |
|                                     |
| Dialing directory..D   run script (Go)....G | Clear Screen.....C |
| Send files.....S     Receive files.....R | cOnfigure Minicom..O |
| comm Parameters....P  Add linefeed.....A | Jump to a shell....J |
| Capture on/off.....L  Hangup.....H       | eXit and reset....X |
| send break.....F     initialize Modem...M | Quit with no reset.Q |
| Terminal emulation.T  run Kermit.....K   | Cursor key mode....I |
| lineWrap on/off....W  local Echo on/off..E | Help screen.....Z   |
|                                     | scroll Back.....B   |
|                                     |
|       Select function or press Enter for none.                             |
|                                     |
|       Written by Miquel van Smoorenburg 1992/1993                         |
|                                     |
+=====+
```

Der Hilfsbildschirm

## Installation und Konfiguration

Der folgende Abschnitt richtet sich hauptsächlich an die Systemverwalterin. Als Anwender des minicom-Programms können Sie zum Abschnitt ``Anwählen eines anderen Rechners" springen.

Die ausführbare minicom Datei sollte in einem Verzeichnis des Suchpfades liegen, der in der PATH Umgebungsvariablen eingetragen ist. minicom muß SUID-root laufen. Ein zusätzliches Programm zur Interpretation von Scriptdateien (runscript) und bei Version 1.4x und früher das Programm keyserv werden im Verzeichnis /etc erwartet. (Die Lokalisierung dieser Dateien wird vor dem Übersetzen der Quelltexte festgelegt und kann in verschiedenen Distributionen variieren.)

In die Datei /etc/termcap sollte ein spezieller Eintrag zur Terminalinitialisierung unter minicom eingefügt werden, mit dem die Darstellung der IBM-Grafikzeichen auf der Console möglich wird:

```
mc|minicom|mc80x25|termcap entry for minicom on the console:\
:is=\E(U\E[m\E>\E[4;20l:\E[?8;25h\E[?1;5;6;7l:\
:rs=\E(B\E[m\E>\E[4;20l:\E[?7;8;25h\E[?1;5;6l:\
:bc=:as=:ae=:am=:vb=\E(B\007\E(U:\
:tc=console:
```

Die Verwendung dieses Eintrags kann mit der Kommandozeilenoption `-t mc` erreicht werden. Um die Verwendung dieser und anderer Kommandozeilenoptionen für alle Anwender vorzubestimmen, kann die Umgebungsvariable MINICOM beispielsweise in der /etc/profile Datei mit einer entsprechenden Zeichenkette belegt werden.

Der erste Aufruf von minicom muß durch die Systemverwalterin erfolgen. Dabei ist die Option `-s` (setup) zu verwenden. Auf diese Weise kommt sie sofort in das Konfigurationsmenü und kann die für alle Benutzer gültige Grundeinstellung vornehmen.

=====[configuration]=====

```

| Filenames and paths |
| File transfer protocols |
| Serial port setup |
| Modem and dialing |
| Screen and keyboard |
| Save setup as dfl |
| Save setup as.. |
| Exit |
+=====+

```

## Das Setup- und Konfigurationsmenü

Die einzelnen Menüpunkte können mit den Cursortasten oder mit den vom vi bekannten Tasten j und k angewählt werden, ESC verläßt das Menü. Der letzte Menüpunkt erscheint nur der Systemverwalterin beim Setup, um das Programm direkt zu verlassen.

```

+=====+
| A - Download directory : . |
| B - Upload directory : . |
| C - Script directory : /usr/lib/minicom |
| D - Script program : /etc/runscript |
| E - Kermit program : /usr/bin/kermit -l %l -b %b |
| |
| Change which setting? |
+=====+

```

## Einstellfenster für Dateinamen und Pfade

Die Verzeichnisse für Up- und Download müssen vorhanden sein bzw. angelegt werden. Angaben ohne absoluten Pfad werden vom Heimatverzeichnis des aufrufenden Benutzers aus gesucht. Diese Verzeichnisse werden von den Dateitransferprotokollen benutzt, die daraus Dateien lesen, um sie auf einen anderen Rechner zu kopieren (Upload) oder dorthin Dateien schreiben, die sie von einem anderen Rechner erhalten (Download).

Das Scriptverzeichnis enthält die minicom Scriptdateien, beispielsweise für das automatische Einloggen. Loginname und Paßwort brauchen nicht im Script gespeichert werden, deshalb ist es kein Sicherheitsrisiko, hier ein zentrales Verzeichnis auch für die Loginscripts anzugeben.

Das runscript-Programm wird von minicom zur Ausführung der Scripts aufgerufen. Hier kann auch ein beliebiger andere Interpreter eingetragen werden. Die Syntax der Scriptdateien ändert sich natürlich entsprechend. Die Kanäle für Standardeingabe und Standardausgabe sind während der Bearbeitung des Scripts mit dem Modem verbunden, die Fehlerausgabe landet auf dem Monitor.

Mit ^A K kann kermit lokal als Client gestartet werden, nachdem auf der Gegenseite ein kermit-Server die Kontrolle über die serielle Schnittstelle übernommen hat. kermit ist ein klassisches Terminalprogramm mit integriertem Protokoll zum Dateitransfer.

Im File transfer protocols Menü werden die Programme zur protokollierten Datenübertragung für den direkten Aufruf aus minicom konfiguriert.

Neben den Bekannten X- Y- Z-Modem Programmen, die unter Unix als Freie Software unter dem Namen rz/sz erhältlich sind, ist in dem Beispiel oben noch kermit als Client zur Datenübertragung aufgeführt.

```

+=====+
| Name          Program          Need name Up/Down FullScr IO-Red. |
| A  zmodem      /usr/bin/sz -vv   Y         U         N         Y         |
| B  ymodem      /usr/bin/sb -vv   Y         U         N         Y         |
| C  xmodem      /usr/bin/sx -vv   Y         U         N         Y         |
| D  zmodem      /usr/bin/rz -vv   N         D         N         Y         |
| E  ymodem      /usr/bin/rb -vv   N         D         N         Y         |
| F  xmodem      /usr/bin/rx -vv   Y         D         N         Y         |
| G  kermit      /usr/bin/kermit -i -l %l -s Y         U         Y         N         |
| H  kermit      /usr/bin/kermit -i -l %l -r N         D         Y         N         |
| I  -           -                 -         -         -         -         |
| J  -           -                 -         -         -         -         |
| K  -           -                 -         -         -         -         |
| L  -           -                 -         -         -         -         |
| |
| Change which setting? (SPACE to delete) |
+=====+

```

Das Fenster zum Einstellen der Datenübertragungsprotokolle

Die Gerätedatei, über die `minicom` das Modem anspricht, sollte immer ein `cua?` Device sein. Die `ttyS?` Devices sind für ankommende Verbindungen vorgesehen. Gelegentlich wird wegen der größeren Aussagekraft ein (symbolischer) Link mit dem Namen `/dev/modem` angelegt, der auf das `cua?` Device zeigt, an dem das Modem angeschlossen ist. Diese Praxis birgt die Gefahr, den Mechanismus des Device-Locking durcheinanderzubringen, indem einige Kommunikationsprogramme für `/dev/cua?` konfiguriert sind, andere für `/dev/modem`. Hier liegt es allein in der Verantwortung der Systemverwalterin, für konsistente Nomenklatur zu sorgen.

```
+=====+
| A -   Serial Device : /dev/modem      |
| B - Lockfile Location: /usr/spool/uucp |
| C -   Callin Program :                |
| D -   Callout Program :               |
| E -   Baud/Par/Bits : 38400 8N1       |
|                                     |
|   Change which setting?              |
+=====+
```

Die Auswahl der seriellen Schnittstelle

Das Verzeichnis, in dem die Lockfiles angelegt werden, muß ebenso wie das Format, in dem diese Dateien vorliegen für alle Kommunikationsprogramme konsistent gehandhabt werden. `minicom` legt seine Prozeß-ID im ASCII Format ab.

Mit Callin/Callout können Programme bestimmt werden, die den Modemport zum Ein- bzw. Auswählen vorbereiten (beispielsweise ein `uugetty` beenden). Unter Linux sollten solche Methoden zum Device-Sharing nicht verwendet werden.


Weitere Informationen über das Device-Handling finden Sie im Abschnitt über [getty](#).

Die Leitungsparameter können über ein weiteres Menü eingestellt werden.

Die Übertragungsgeschwindigkeit ist vom angeschlossenen Modem abhängig. Moderne Geräte mit Datenkompression (MNP5 oder V42bis) werden auf der Modem-Rechner-Verbindung mit einer festen Baudrate angesteuert, die höher sein sollte als der höchste anzunehmende Durchsatz der eigentlichen Modemverbindung. Bei 2400-Baud-Modems reichen 9600 bps in der Regel aus.

```
+=====[Comm Parameters]=====+
| Current: 38400 8N1                |
|                                     |
|   Speed          Parity          Data |
| A: 300           H: None          M: 5 |
| B: 1200          I: Even          N: 6 |
| C: 2400          J: Odd           O: 7 |
| D: 4800                          P: 8 |
| E: 9600                          |
| F: 19200         K: 8-N-1         |
| G: 38400         L: 7-E-1         |
|                                     |
| Choice, or <Enter> to exit?       |
+=====+
```

Die Einstellung der seriellen Schnittstelle

Die maximale Übertragungsrate kann mit dem `setserial`-Programm von 38400 bps auf 57600  bps gesetzt werden.

Ältere Modems ohne automatische Baudratenerkennung senden (und empfangen) die Daten mit genau der Übertragungsrate, mit der sie mit dem Rechner verbunden sind. Hier muß also die Einstellung der realen Modemverbindung angepaßt werden.

Die Einstellungen für Bytelänge, Parität und Stopbits sind fast immer 8-N-1 (8 Bit/Byte, keine Parität, ein Stopbit).

Weitere Einstellungen können zur Laufzeit nicht vorgenommen werden. Das RTS/CTS Hardware-Handshaking wird immer automatisch eingeschaltet.

```
+=====[Modem and dialing parameter setup]=====+
```

```
A - Init string ..... ^MATZ^M
B - Reset string ..... ^M~ATZ^M~
C - Dialing prefix #1.... ATDP
D - Dialing suffix #1.... ^M
E - Dialing prefix #2.... ATDP
F - Dialing suffix #2.... ^M
G - Dialing prefix #3.... ATX1DP
H - Dialing suffix #3.... ;X4D^M
I - Connect string ..... CONNECT
J - No connect strings .. NO CARRIER          BUSY
                        NO DIALTONE           VOICE
K - Hang-up string ..... ~~+++~~ATH^M
L - Dial cancel string .. ^M

M - Dial time ..... 45      P - Auto baud detect .... Yes
N - Delay before redial . 60  Q - Drop DTR to hangup .. Yes
O - Number of tries ..... 10 R - Modem has DCD line .. Yes

Change which setting?      (Return or Esc to exit)
+=====+
```

Die Einstellung der Modembefehle

Bei den Hayes-kompatiblen Modems sind die hier dargestellten Steuerbefehle normalerweise ausreichend. In Spezialfällen müssen die entsprechenden Sequenzen dem Modemhandbuch entnommen werden.

Wenn das Modem im Auto-Answer-Mode betrieben wird, um die Einwahl in das System zu erlauben, ist es sinnvoll, die DTR-Leitung beim Auflegen „Low`` zu setzen (Drop DTR to hangup.. YES), um sicherzustellen, daß das Modem erst Anrufe entgegennimmt, wenn das getty wieder ein DTR-Signal gibt.

```
+=====[Screen and keyboard]=====+
|
| A - Command key is      : ^A
| B - Backspace key sends : DEL
| C - Status line is     : enabled
|
| Change which setting?   (Esc to exit)
|
+=====+
```

Bildschirm- und Tastatureinstellungen

Von der in diesem Menü angebotenen Möglichkeit, die Tastenkombination für die minicom-Kommandos von CONTROL-A auf die ALT-Taste zu legen, sollte bei der deutschen Tastatur kein Gebrauch gemacht werden. Der Tastaturtreiber produziert hier Codes, die von MINICOM nicht unbedingt korrekt interpretiert werden. Das kann zu einer Situation führen, aus der nur ein kill(1) wieder herausführt.

Wenn Sie als Superuserin alle Einstellungen für Ihr System eingegeben haben, können Sie diese Konfiguration als Default für alle Anwender speichern (Save setup as dfl im Konfigurationsmenü).

WICHTIG!

Um den normalen Benutzern den Aufruf von minicom zu erlauben, müssen Sie die Benutzernamen in der Datei /etc/minicom.users eintragen.

Anwählen eines anderen Rechners

Das Terminalprogramm minicom verwaltet eine Telefonliste im Heimatverzeichnis eines jeden Benutzers. Diese Liste kann im Dial-Menü interaktiv bearbeitet werden. In dieses Menü gelangen Sie durch das Tastenkommando ^D RETURN.

```
+-----[Dialing Directory]-----+
|      Name      Number      Line Format Terminal Script      |
|  1  LunetIX    030 6235097   38400 8N1    VT100   unixlogin  |
```

2	INTERW.	030 2513771	38400 8N1	VT100	unixlogin
3	Access	030 2513771	38400 8N1	VT100	unixlogin

( Press Escape to exit.. )

## Das Wählmenü

Mit den Cursortasten oder den vom `vi` Editor bekannten Tasten `h`, `j`, `k`, und `l` kann im oberen Fenster ein Eintrag in der Telefonliste ausgewählt (hoch/runter) und im unteren Fenster eine Aktion für diese Telefonnummer bestimmt werden (links/rechts). Die Aktion wird ausgeführt, sobald die Wahl mit `RETURN` bestätigt ist.

Mit `Add` wird ein leerer Eintrag unter dem aktuellen angelegt. Mit `Edit` kann der aktuelle Eintrag verändert werden. Dazu erscheint die folgende Maske:

```

+=====+
| A -   Name           : LunetIX
| B -   Number         : 030 6235097
| C -   Dial string #   : 1
| D -   Local echo      : No
| E -   Script          : unixlogin
| F -   Username        : she
| G -   Password        : <Passwort>
| H -   Terminal Emulation : VT100
| I -   Line Settings   : 38400 8N1
|
|      Change which setting?
+=====+
```

## Eingabefenster für einen Eintrag im Wählmenü

Die meisten Einträge dieser Maske haben aussagekräftige Namen.

Der `Dial string` wird im „Modems and dialing“ Setup-Menü festgelegt. Die Systemverwalterin sollte die Grundeinstellung von `minicom` so durchgeführt haben, daß Sie hier immer mit dem `` `Dial String #1`“ auskommen dürften. `Local echo` ist eine Einstellung für Halb-Duplex-Verbindungen (ein Relikt aus dem Mittelalter der Datenfernübertragung).

Das Script wird unmittelbar nach erfolgreichem Verbindungsaufbau abgearbeitet (Login-Script). Die Einträge `Username` und `Password` können im Login-Script verwendet werden.

Wenn Sie einen Eintrag des Wählmenüs mit `Dial` bestätigen, wählt das Modem automatisch die Telefonnummer und versucht eine Datenverbindung herzustellen. Ist dieser Versuch erfolgreich, wird die `login:` Meldung der gegenstelle an die lokale `minicom`-Terminalemulation gesendet. In der Beispielkonfiguration oben wird diese

Meldung automatisch durch das `unixlogin` Script beantwortet. Sie können diese Prozedur natürlich auch von Hand erledigen.

Nachdem Sie sich in dem fremden Rechner eingeloggt haben, können Sie dort so arbeiten wie an Ihrem Linux-Rechner.

## Automatische Frage- und Antwort-Spiele

Mit dem `runscript`-Programm kann `minicom` Scriptdateien ablaufen lassen. Die Standardkanäle von `runscript` werden mit der seriellen Schnittstelle verbunden. Auf diese Weise können automatische Dialoge abgewickelt werden, beispielsweise zum Einloggen in den angewählten Rechner (remote host).

In einer Scriptdatei für `runscript` können folgende Befehle verwendet werden:

`send` ***String***

Die Zeichenkette *String* wird in den Standardkanal geschrieben. Wenn ein einzelnes Wort gesendet werden soll, braucht es nicht in Anführungszeichen eingeschlossen werden, sonst muß die Zeichenkette in Anführungszeichen stehen. Es werden die üblichen Escape-Sequenzen akzeptiert:

`\a`

Alarm (Piep)

`\b`

Schritt zurück

`\c`

kein Zeilenende

`\f`

Seitenvorschub

`\n`

Zeilenende

`\r`

Wagenrücklauf

`\nnn`

das Zeichen mit dem (oktalen) Code *nnn*

Außerdem können alle Umgebungsvariablen in der Form `$(VARIABLE)` eingefügt werden. Zwei neue Variable werden von `minicom` mit den entsprechenden Werten belegt, wenn diese in der Telefonliste angegeben sind. In `$(LOGIN)` steht der Loginname und in `$(PASS)` das Paßwort für die aktuelle Telefonverbindung.

`print` ***String***

Im Unterschied zu `send` schreibt `print` die Zeichenkette *String* in den Fehlerkanal, der mit dem lokalen Bildschirm verbunden ist.

***Label:***

Mit dem *Label* wird eine Zeile markiert, in die mit einer `goto` oder `gosub` Anweisung gesprungen werden soll.

`goto` ***Label***

Das Script-Programm wird in der mit *Label* markierten Zeile fortgesetzt.

`gosub` ***Label***

Im Unterschied zum Sprung mit `goto` kann beim `gosub`-Sprung durch eine `return` Anweisung an die Absprungstelle zurückgekehrt werden. `gosub`-Aufrufe dürfen verschachtelt werden.

`return`

Die Subroutine wird beendet und das Programm mit der Zeile nach dem `gosub` fortgesetzt, das diese Subroutine aufgerufen hat.

## ! **Kommando**

Das externe *Kommando* wird ausgeführt. Der Status wird in der Umgebungsvariablen \$? zurückgeliefert und kann in einer `if` Abfrage verwendet werden.

## exit [**Wert**]

Die Scriptbearbeitung wird abgebrochen und der angegebene *Wert* als Status weitergegeben.

## set **Variable Wert**

Die *Variable* wird mit dem *Wert* belegt. Variablennamen können nur die Buchstaben a-z sein. Als Werte sind sowohl ganze Zahlen als auch Zeichenketten erlaubt.

## inc **Variable**

Die *Variable* wird um eins erhöht.

## dec **Variable**

Die *Variable* wird um eins erniedrigt.

## if **Wert Operator Wert Statement**

Das Statement wird ausgeführt, wenn der Wertevergleich wahr ist. Als Operatoren sind `<`, `>` und `=` erlaubt.

## timeout **Wert**

Der Timeout wird auf den *Wert* gesetzt. Voreinstellung sind 120 Sekunden außerhalb einer `expect` Anweisung und 60 Sekunden innerhalb einer `expect`-Anweisung. Wenn der Timeout abgelaufen ist, wird die Bearbeitung des Scripts automatisch abgebrochen.

## verbose {on|off}

Wenn `verbose 'on'` ist, werden alle Zeichen, die vom Standardkanal gelesen werden, im Fehlerkanal (also auf dem Bildschirm) dargestellt.

## sleep **Wert**

Die Ausführung des Scripts wird die angegebene Anzahl Sekunden angehalten.

## expect { **Muster** [**Statement**]}...

Es werden so lange Zeichen aus dem Eingabekanal gelesen, bis eines der Muster darin gefunden wird. Dann wird entweder das dazu gehörende Statement ausgeführt, oder, wenn keins vorhanden ist, die `expect` Anweisung verlassen.

Das Muster ist eine Zeichenkette wie bei `send`. Im Muster können keine regulären Ausdrücke verwendet werden.

## break

Zu einer `timeout`-Zuweisung innerhalb einer `expect`-Anweisung kann das Statement `break` angegeben werden, um den Abbruch der Scriptbearbeitung zu verhindern und die Fortsetzung nach der `expect` Anweisung zu erreichen.

## call **Scriptdatei**

Die Ausführung verzweigt zur angegebenen Scriptdatei. Wenn diese Datei fehlerfrei ausgeführt wurde, wird die aktuelle Scriptdatei mit der auf diese Anweisung folgenden Zeile fortgesetzt.

Das folgende **Beispiel** ist nicht nur geeignet, die Benutzung von `runscript` zu veranschaulichen, es ist überhaupt das Standardscript zum Einloggen in alle Linux-Systeme.

```
# Standard UNIX Loginscript.
# Mit diesem Script von Miquel van Smoorenburg kann das Einloggen
# in (fast) alle Linux{\bindestrich} und UNIX{\bindestrich}Boxen automatisiert werden.
#
# Einige Variable
set a 0
set b a
print Trying to Login..
# Irgendjemand hat wohl festgestellt, daß das erste send
# manchmal besser übersprungen wird.
goto skip
loop1:
```



```

# Der Loginname wird höchstens dreimal gesendet.
send ""
inc a

skip:
if a > 3 goto failed1
expect {
    "ogin:"
    "ssword:"      send ""
    "NO CARRIER"   exit
    timeout 2       goto loop1
}

loop2:
send "${LOGIN}"

# Das Passwort wird auch nicht mehr als dreimal gesendet.
inc b
if b > 3 goto failed1
expect {
    "ssword:"
    "ogin:"      goto loop2
    timeout 2    goto loop2
}
send "${PASS}"
# Wenn nicht innerhalb von 3 Sekunden ein "incorrect" ankommt,
# war das Passwort wohl OK. Wenn nach dem Terminal gefragt
# wird, stellen wir vt100 ein.
expect {
    "TERM="      send "vt100"
    "incorrect"   goto loop1
    timeout 3     break
}
exit
failed1:
print \nLogin fehlgeschlagen (Falsches Passwort?)
exit

```

## Dateien übertragen

Mit einer Terminalemulation wie `minicom` kann auf einem entfernten Rechner gearbeitet werden wie auf dem lokalen. Wenn die Verbindung mit einem Hochgeschwindigkeitsmodem hergestellt wird, kann an der Bildschirmausgabe kein Unterschied zwischen Local- und Remote-Host festgestellt werden.

Allerdings können alle Arbeitsergebnisse, die als Dateien vorliegen, nicht ohne weiteres auf dem lokalen Rechner weiterverarbeitet werden. Bei einem echten vt100-Terminal macht das auch keinen Sinn, weil so ein Terminal eben nur Daten auf dem Bildschirm anzeigen bzw. von der Tastatur lesen kann. Bei einem Linux-Rechner, der ein vt100-Terminal emuliert, ist das eine ernste Einschränkung.

Dateien, die nur druckbare Zeichen enthalten, können theoretisch durch einfache Ausgabeumlenkung in einer Datei gespeichert werden. Der `Capture` Buffer bietet die Möglichkeit, eine Mitschrift einer kompletten Terminalsitzung abzuspeichern. So eine Mitschrift kann dann anschließend bearbeitet werden, indem beispielsweise einzelne Teile in neue Textdateien kopiert werden.

Viele Dateien enthalten aber nichtdruckbare Zeichen. Um solche Dateien übertragen zu können, werden spezielle Programme benötigt, die für die Dauer der Übertragung den Datenstrom umlenken.

Um auch bei alten Modems ohne automatische Fehlerkorrektur einen zuverlässigen Dateitransfer zu gewährleisten, benutzen die Übertragungsprogramme Protokolle mit Fehlererkennung. Die Datei wird nicht als Ganzes übertragen, sondern in kleinen Stücken. Jedes Stück wird gemeinsam mit einer Checksumme (Cyclic Redundancy Check, CRC) gesendet, so daß der Empfänger die korrekte Übertragung nachprüfen kann. Im Fehlerfall wird das Paket noch einmal angefordert.

Ein beliebtes Transferprotokoll ist mittlerweile in der dritten Generation verbreitet. Die Protokolle sind als X- Y- und Z-Modem bekannt. Die Kommandos `rx`, `rb` und `rz` für den Empfang bzw. `sx`, `sb` und `sz` für das Senden implementieren diese Protokolle unter Linux.

Die Y- und Z-Modem-Protokolle übertragen den Dateinamen gemeinsam mit der eigentlichen Datei. Von den Empfangsprogrammen benötigt deshalb nur `rx` (X-Modem) einen Dateinamen auf der Kommandozeile. Das Z-Modem-Protokoll verringert die Paketgröße bei hoher Fehlerrate, um den Overhead durch mehrfach angeforderte Pakete zu minimieren. Außerdem werden ununterbrochen Pakete ohne Bestätigung des Empfängers (Acknowledgement, ACK) übertragen. Erst wenn ein Fehler erkannt wird, unterbricht das empfangende Programm den Datenfluß und fordert das fehlerhafte Paket nochmal an. Das Z-Modem-Protokoll zeigt die höchste Übertragungsrate aller Protokolle.

Das `kermitt`-Kommunikationsprogramm benutzt ein eigenes Protokoll zum Dateitransfer. In der hier dargestellten Form wird das Programm so aufgerufen, daß es nur den Dateitransfer durchführt. Auch hier ist wie beim Aufruf des interaktiven Programms darauf zu achten, daß eine `.kermrc`-Datei im Heimatverzeichnis die korrekten Startparameter setzt.

Die `rzsz`-Programme unter Linux lesen aus der Umgebungsvariable `RZSZLINE` den Port, auf dem die Übertragung laufen soll (beispielsweise `/dev/cua1`). Um eine Dateiübertragung vom Remote- zum Localhost zu starten, wird zuerst auf dem anderen Rechner ein Befehl wie der folgende eingegeben:

```
$ sz foo.bar.tgz
**B000000000000000
```

Dadurch wird das `sz` (send z-modem) Programm für die Datei `foo.bar.tgz` gestartet. Die „Antwort“ `**B000000000000000` kann von anderen Terminalemulationen für den automatischen Start eines Z-Modem Empfangs genutzt werden. Im Fall von `minicom` muß der Dateitransfer auch auf dem lokalen Rechner manuell gestartet werden. Dazu wird nach dem Tastaturkommando `^A r` (receive) das Z-Modem-Protokoll ausgewählt. Der Fortschritt der Übertragung kann dann in einem kleinen Bildschirmfenster beobachtet werden. Die empfangene Datei wird im [Downloadverzeichnis](#) abgelegt.

## term

Der `term`-Server von Michael O'Reilly und die dazugehörigen Clients `trsh`, `tupload`, `tredir`, `txconn` und `tmon` erlauben die gleichzeitige Benutzung mehrerer logischer Datenkanäle auf einer seriellen Leitung. Die ein wenig an SLIP erinnernde Funktionalität wird vollkommen ohne Kernelunterstützung und ohne privilegierte Ausführungsrechte erreicht.

An jedem Ende übernimmt jeweils ein `term`-Server die vollständige Kontrolle über den Datenfluß auf der seriellen Verbindung. Die lokalen Clients verbinden sich über ein Unix-Domain-Socket mit dem lokalen `term`-Server. Die beiden Server tauschen Datenpakete aus und bedienen so die Clients auf beiden Seiten. Die beiden Server arbeiten gleichberechtigt, es können von beiden Seiten mehrere Benutzer die Verbindung nutzen.

## Kurzer Überblick

Der `term`-Server ist nicht in der Lage, selbst eine Verbindung zu einem anderen Rechner aufzubauen. Um eine `term`-Session zu starten, muß zuerst eine normale Terminalverbindung hergestellt werden. Dann wird der `term`-Server auf dem Remote-Host (dem anderen Rechner) im Vordergrund gestartet, indem auf der Kommandozeile das `term`-Programm aufgerufen wird. Daraufhin übernimmt `term` die Kontrolle über „seinen“ Port. Der Shellprompt wird nicht wieder ausgegeben.

Ohne das Terminalprogramm zu beenden, muß daraufhin auf dem lokalen Rechner der zweite `term`-Server gestartet werden. Hierbei müssen die Standardkanäle mit dem Modemdevice verbunden werden. Die meisten Terminalprogramme bieten auf die eine oder andere Weise die Möglichkeit, lokale Kommandos mit dieser Umleitung aufzurufen. Hierbei muß aber darauf geachtet werden, daß das Terminalprogramm tatsächlich die Schnittstelle transparent macht, also alle Zeichen aus beiden Richtungen unverändert weiterleitet.

Es ist aber möglich, von einer anderen Shell aus das `term`-Programm im Hintergrund zu starten, indem beispielsweise die folgende Kommandozeile eingegeben wird:

```
$ term -v /dev/cua1 2> /dev/null &
```

Wenn beide Server gestartet sind, kann von jeder Shell mit den geeigneten Benutzerrechten ein `term`-Client gestartet werden.

Mit dem `trsh`-Client kann eine interaktive Shell auf dem Remote-Host gestartet werden (wie `rlogin`). Ebenso gut kann auch ein einzelnes Kommando `remote` ausgeführt werden.

Mit `tredir` können beliebige IP-Ports des lokalen Systems auf Ports des Remote-Host umgelenkt werden. Auf diese Weise können Internetdienste des Remote-Host auch auf dem lokalen Rechner benutzt werden.

Mit `tupload` können Dateien über die serielle Verbindung kopiert werden.

Der `txconn`-Client wird auf dem Remote-Host gestartet und richtet ein logisches X11-Display ein. Alle X11-Clients die dieses Display benutzen, werden mit dem lokalen X11-Server verbunden.

## Der `term` Server

Im Unterschied zur einfachen Terminalemulation, die keine besonderen Anforderungen an den Remote-Host stellt, muß der `term`-Server auf beiden Rechnern installiert sein. Sowohl der Server als auch die Clients arbeiten mit normalen Benutzerrechten. Die Programme stehen unter der GNU General Public License und sind deshalb als Quelltexte erhältlich. Es ist möglich, mit einem normalen Benutzeraccount das `term`-Paket zu installieren, vorausgesetzt, es ist ein C-Compiler auf dem System installiert und zugänglich. Die Quelltexte sind für verschiedene Systeme vorbereitet. Ohne Garantie und Anspruch auf Vollständigkeit sind das NeXT, AIX, BSD, SGI, SVR4, Solaris 2.1 und mit Einschränkung DEC Ultrix mit MIPS Compiler sowie HP-UX. Wenn es Probleme beim Übersetzen gibt, kann der SysOp vielleicht weiterhelfen.

Die fertigen Programme müssen in einem Verzeichnis des Suchpfades (`PATH`) installiert werden. Das kann beispielsweise `/usr/local/bin` sein. Das Verzeichnis `~/bin` geht ebenfalls. Außerdem braucht `term` das Verzeichnis `~/.term` im Heimatverzeichnis des Benutzers. Wenn es nicht existiert, wird es beim ersten Aufruf automatisch erzeugt.

Um die einwandfreie Funktion des Servers zu erreichen, sollte in diesem Verzeichnis eine Konfigurationsdatei namens `termrc` angelegt werden. In dieser Datei können die folgenden Einstellungen vorgenommen werden (zwischen Option und Argument muß immer genau ein Leerzeichen stehen; leere Zeilen und solche, die mit einem `#` beginnen, werden ignoriert):

`escape` *Nummer*[-*Nummer*]

veranlaßt `term`, das angegebene Zeichen bzw. den Bereich von Zeichen nicht zu senden. Die Zeichen und Bereiche müssen vor dem ersten Start von `term` mit dem `linecheck`-Programm ermittelt werden.

`ignore` *Nummer*[-*Nummer*]

ist die zu `escape` korrespondierende Einstellung für die Gegenseite. Alle Zeichen und Bereiche, die auf der einen Seite mit `escape` ausgeblendet werden, sollten auf der anderen Seite mit `ignore` markiert sein.

`compress` {on|off}

schaltet die Datenkompression ein bzw. aus. Wenn bereits komprimierte Dateien gesendet werden oder die Modemverbindung mit einem komprimierenden Protokoll arbeitet, ist die zusätzliche Kompression durch `term` nicht sinnvoll. Die Kompression kann mit der `-c` Option für jeden Client einzeln ab- und eingeschaltet werden.

`baudrate` *Geschwindigkeit*

begrenzt die Geschwindigkeit, in der `term` Daten über die serielle Schnittstelle schickt. Diese Einstellung wird nicht auf den Schnittstellentreiber im Kernel übertragen, sie dient nur dem internen Timing von `term`. Der Wert sollte nicht größer als die reale Übertragungsrate gewählt werden.

`timeout` *Zeit*

bestimmt die Zeitspanne in 20stel Sekunden, die `term` nach dem Senden eines Pakets auf die Bestätigung der

Gegenstelle wartet. Nach dieser Zeit wird das Paket automatisch nochmal gesendet.

window **Größe**

bestimmt die Anzahl der Datenpakete, die „blind“ gesendet werden können. Ein weiteres Paket wird erst gesendet, nachdem das erste bereits gesendete Paket bestätigt wurde. Voreinstellung ist 3.

noise on

Wenn dieser Schalter gesetzt ist, schreibt term alle Daten, die es nicht versteht, in die Fehlerausgabe. Auf diese Weise kann auf dem lokalen Rechner beispielsweise ein talk-Angebot oder eine Systemmeldung vom Remote-Host empfangen werden.

remote

bestimmt diese Instanz von term zum Remote-Server. Dieser Eintrag gehört in die termrc auf dem fernen Rechner.

sevenbit

muß auf 7-Bit Datenleitungen anstelle von escape 128-255 verwendet werden.

seven\_in

7-Bit auf der ankommenden Leitung

seven\_out

7-Bit auf der rausgehenden Leitung

shift **Zahl**

Alle Zeichen werden mit der angegebenen Zahl (0-255) XOR verknüpft. Auf diese Weise können bestimmte Zeichensequenzen aus intransparenten Übertragungsbereichen herausgeholt werden.

breakout **Nummer**

bestimmt das Zeichen mit der angegebenen Nummer zum „breakout“. Wenn fünf dieser Zeichen in der Standardeingabe erscheinen, wird der term-Server beendet.

chdir **Verzeichnis**

bestimmt das Verzeichnis, in dem die Kommandos für trsh starten. Relative Pfade für tupload werden auf dieses Verzeichnis bezogen.

denytrsh on

veranlaßt term, jede trsh-Verbindung von der Gegenseite abzulehnen.

chroot **Verzeichnis**

veranlaßt term, in einer chroot(2)-Umgebung zu starten.

Das folgende **Beispiel** kann als Ausgangspunkt für eigene Versuche dienen:

```
# Beispiel einer ~/.term/termrc-Datei
```

```
compress off
```

```
# Für ein Modem mit Datenkompression brauchen die Daten von term  
# nicht komprimiert zu werden.
```

```
escape 0-31
```

```
escape 128-159
```

```
# Diese Einstellung wird für die ersten Versuche empfohlen.  
# Für eine Linux<->Linux-Verbindung brauchen normalerweise keine  
# Zeichen ausgeschlossen zu werden
```

```
baudrate 38400
```

```
# Diese Übertragungsrate wird nur bei Textdateien erreicht.  
# Vorkomprimierte Daten können zu Problemen führen, wenn das timeout  
# nicht entsprechend hoch eingestellt ist.
```

```
timeout 100
```

```
# (100/20 = 5 Sekunden) ist für Modems ohne Kompression und Korrektur  
# besser mit 50 anzusetzen.  
# Bei komprimierenden Modems muß das timeout die Verzögerung zwischen  
# senden der Daten in den Modempuffer und Übertragung der Daten mit
```

```
# anschließender Bestätigung überbrücken.

window 5
# Es werden bis zu fünf Pakete ohne Bestätigung vorausgeschickt.
# Sinnvolle Werte liegen zwischen 2 und 6, bei sehr schnellen
# Modems auch höher.

#shift 224
# Die ersten 32 Zeichen werden mit dem Bereich über 224
# vertauscht, die anderen werden ein wenig durcheinandergewürfelt.

#flowcontrol 500
# Alle 500 Zeichen wird ein Control-Q gesendet, damit ein
# versehentlich durch Control-S angehaltener Datenstrom wieder
# in Fluß kommt.

noise on
# Alle Daten (Pakete), deren Checksumme nicht stimmt, werden in den
# Standardfehlerkanal geschrieben.

#sevenbit
# Für den Fall, daß linecheck eine 7-Bit-Leitung feststellt.

#seven_in
# Wenn nur eine Richtung mit 7-Bit arbeitet, kann dem mit seven_in/
# seven_out Rechnung getragen werden.

#seven_out
# Die symmetrische Einstellung für die "andere Seite"

#ignore 7
# Ein ankommendes Klingelzeichen (ASCII 7) wird ignoriert.

#breakout 24
# Fünfmal ^X (Control-X) veranlaßt den term-Server, zu terminieren.

# chdir /home/she
# Alle trsh-Programme werden in meinem Heimatverzeichnis gestartet.
```

```
# remote
# Mit dieser Einstellung wird einer der beiden Server zum Slave
```

Die folgenden Umgebungsvariablen können benutzt werden:

#### BAUDRATE

wird benutzt, wenn auf der Kommandozeile und in der Konfigurationsdatei keine anderen Werte bestimmt sind.

#### SHELL

bestimmt die Shell, die von trsh gestartet wird.

#### DISPLAY

wird benutzt, um den X11-Server für txconn zu finden.

#### TERMDIR

kann ein Verzeichnis bestimmen, in dem das .term-Verzeichnis mit dem Socket angelegt wird.  
Voreinstellung ist \$ ( HOME ).

Folgende Kommandozeilenoptionen werden erkannt:

#### -s *Geschwindigkeit*

setzt die maximale Übertragungsgeschwindigkeit

#### -n {on|off}

entspricht der *noise* Einstellung in der Datei termrc

#### -c {on|off}

schaltet die Datenkompression an bzw. ab

-r

bestimmt den Remote-Server

-f **Anzahl**

veranlaßt `term`, alle *Anzahl* Zeichen ein XON zwecks Flow-Control zu senden

-w **Größe**

setzt die Größe des Übertragungsfensters

-t **Timeout**

setzt das Timeout (in 20stel Sekunden)

-o

erzwingt das dauernde Senden von Paketen; wenn keine neuen Daten anliegen, wird das letzte Paket wiederholt

-a

schaltet den 7-Bit-Modus ein

-d **Level**

bestimmt den Debuglevel

-l **Datei**

bestimmt die Datei zum Logfile für Debugmeldungen und Rauschen

-v **Device**

veranlaßt `term`, auf der angegebenen Gerätedatei zu arbeiten.

-l

veranlaßt `term`, den Standardausgabekanal anstelle des Standardeingabekanals als Modemport zu verwenden; zusammen mit `-v` wird diese Option ignoriert

## Leitungstransparenz

`term` arbeitet auf bestehenden seriellen Verbindungen. Die Einstellungen der Schnittstellentreiber werden weitgehend von dem Terminalprogramm übernommen, mit dem der `term`-Server gestartet wird. Einige Einstellungen des Treibers können zu Veränderungen des Datenstroms führen. Beispielsweise können bestimmte Steuerzeichen „verschluckt“ oder „übersetzt“ werden. Auf diese Weise entstehen Lücken im Übertragungsspektrum. Weil `term` darauf angewiesen ist, daß alle Daten, die es sendet, auch genau so ankommen, wie sie abgeschickt wurden, müssen die intransparenten Bereiche aus dem Übertragungsspektrum ausgeblendet werden. Diese Zeichen werden dann als Escape-Sequenzen in transparente Bereiche eingeblendet.

Die Überprüfung der Leitungstransparenz kann nicht von `term` selbst durchgeführt werden. Die notwendigen Informationen müssen vom Anwender in der `termrc`-Datei abgelegt werden.

Um eine Verbindung zu prüfen, ist im `term`-Paket das `linecheck`-Programm enthalten. Dieses Programm muß auf beiden Seiten der Leitung mit der gleichen Umlenkung der Standardkanäle gestartet werden wie der `term`-Server. Die Fehlerausgabe muß in einer Datei gesichert werden. Das geschieht indem auf dem Remote-Host das folgende Kommando eingegeben wird:

```
$ linecheck 2> /tmp/linecheck.log
```

auf dem lokalen Rechner wird anschließend das folgende Kommando eingegeben (darauf achten, daß das Terminalprogramm nicht nebenherläuft):

```
$ linecheck </dev/cua1 >/dev/cua1 2> /tmp/linecheck.log
```

Wenn beide Programme korrekt gestartet wurden, nehmen sie in einer ersten „Handshaking“-Phase Kontakt miteinander auf und senden sich dann gegenseitig in einer festgelegten Reihenfolge alle 256 möglichen Bytes zu bzw. bestätigen deren Empfang. Das dauert ziemlich lange, weil nur ein Zeichen pro Sekunde gesendet wird. Wenn die Übertragungssequenz abgeschlossen ist, geben die beiden Programme jeweils eine Zusammenfassung der

gefundenen Übertragungsfehler. Mit dieser Information lassen sich dann die entsprechenden `escape` Bereiche für die andere `termrc`-Datei bestimmen.

## trsh

Syntax: **trsh** *[-rc] [-t Server] [-p Priorität] [[-s] Kommando]*

Ohne weitere Argumente startet `trsh` eine Shell auf dem Remote-Host, ähnlich wie `rlogin`. Der Remote-term-Server belegt dazu ein Pseudoterminal (pty).

Wenn ein Kommando angegeben ist, wird dieses Kommando remote ausgeführt. Durch die zusätzliche Option `-s` wird der Remote-Server veranlaßt, für die Kommandoausführung kein Pseudoterminal zu belegen.

Im Unterschied zu `rsh` wird die Fehlerausgabe bei `trsh` mit der Standardausgabe zusammengelegt. Außerdem wird die Verbindung zum Server und damit zum Remote-Kommando sofort unterbrochen, wenn `trsh` ein EOF (CONTROL-D) in der Standardeingabe findet.

## tupload

Syntax: **tupload** *[-rcfuvq] [-t Server] [-p Priorität] [-as Dateiname] Datei ...[Verzeichnis]*

`tupload` kopiert eine oder mehrere Dateien von dem System, auf dem es aufgerufen wird, auf das andere. Wenn als letztes Argument ein Verzeichnis auf dem Zielrechner benannt ist, werden die Dateien dorthin kopiert.

Wenn eine Datei bereits auf dem Zielrechner existiert, wird automatisch getestet, ob sie kleiner ist als die lokale Datei. In diesem Fall wird davon ausgegangen, daß ein früherer Upload fehlgeschlagen ist, und die Datei wird vervollständigt.

`tupload` erlaubt folgende Kommandozeilenoptionen:

`-f`

(force) existierende Dateien werden überschrieben

`-u`

(unlink) nach erfolgreicher Übertragung werden die Quelldateien gelöscht

`-v`

(verbose) gibt die Anzahl der übertragenen Bytes, den Namen der Zieldatei und die Übertragungsgeschwindigkeit aus

`-q`

(quiet) unterdrückt alle Meldungen

`-as Dateiname`

zu jeder Quelldatei kann mit dieser Option ein Name für die Zieldatei festgelegt werden

## tredir

Syntax: **tredir** *[-rc] [-t Server] [-p Priorität] LocalPort [hostname:]RemotePort ...*

`tredir` verbindet lokale TCP-Ports mit Ports auf dem Remote-Host oder auf einem weiteren Internetrechner. Beispielsweise verbindet das Kommando

```
$ tredir 2000 soho.lunetix.de:23
Redirecting 2000 to soho.lunetix.de:23
$
```

den (unbenutzten) lokalen IP-Port Nummer 2000 mit dem Port Nummer 23 (telnet) auf soho.lunetix.de (das ist nicht unbedingt der Remote-Host). Wenn diese Verbindung hergestellt ist, kann mit dem Kommando

```
$ telnet localhost 2000
```

```
Trying 127.0.0.1...
Connected to atlantis.
Escape character is '^['.
```

```
Linux 0.99.12 (soho) (ttyp0)
```

```
soho login: _
```

eine „umgeleitete“ telnet-Verbindung zu soho.lunetix.de hergestellt werden. (Die Ausgabe von telnet zeigt deutlich, daß die „echte“ telnet-Verbindung zum Localhost atlantis besteht. Erst durch die Umleitung wird daraus eine Verbindung mit soho. Auf soho erscheint die Verbindung wiederum so, als sei sie von cicero.lunetix.de (dem Remote-Host) aus hergestellt worden.

Die Portnummern für die *well known services* sind in der Datei `/etc/services` aufgeführt. Durch Verbindung eines unbenutzten lokalen Ports mit einem Serviceport auf einem anderen Rechner können die entsprechenden Services vom lokalen Rechner aus benutzt werden.

Um die Ports 1:1 umlenken zu können, dürfen auf dem lokalen Rechner keine Anwendungen die entsprechenden Ports benutzen. Das bedeutet insbesondere, daß der `inetd` diese Ports nicht überwachen darf. Außerdem dürfen Ports mit Nummern bis 1024 nur mit Rootprivilegien geöffnet werden.

## txconn

Syntax: **txconn** *[-rc] [-t Server] [-p Priorität]*

txconn wird typischerweise einmal zu Anfang einer Session auf dem Remote-Host gestartet. Das Programm setzt sich automatisch in den Hintergrund. Dort stellt es ein logisches Display für alle X11-Clients zur Verfügung, das durch den term-Server auf den lokalen X11-Server umgeleitet wird. Normalerweise erhält das logische Display die Nummer 9. Der „echte“ X11-Server hat meistens die Nummer 0.

Um die X11-Clients auf dem Remote-Host zu veranlassen, mit dem logischen Display von txconn zu arbeiten, kann das Display auf der Kommandozeile angegeben werden:

```
$ xterm -title 'xterm auf cicero' -display cicero.lunetix.de:9 &
$ _
```

oder das Display wird für alle Clients gemeinsam in der Umgebungsvariablen DISPLAY gesetzt.

```
$ export DISPLAY=cicero.lunetix.de:9
$ _
```

## tmon

Syntax: **tmon** *[-rc] [-t Server] [-p Priorität]*

tmon fragt in regelmäßigen Abständen vom lokalen term-Server die Anzahl der Clients und den Datendurchsatz ab und gibt die Werte auf dem Bildschirm aus.

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [Seriell einloggen](#) **Up:** [Datenreisen und reisende Daten](#) **Previous:** [Technische Voraussetzungen](#)



## Subsections

- [Das richtige getty](#)
    - [Device-Handling](#)
    - [Lockfiles](#)
  - [getty\\_ps](#)
    - [Syntax:](#)
    - [Optionen:](#)
    - [Defaults-Dateien](#)
    - [gettydefs](#)
- 

# Seriell einloggen

Mit dem gleichen technischen Equipment, das für eine Terminalsession auf einem anderen Rechner notwendig ist, läßt sich die umgekehrte Verbindung aufbauen. Das Modem kann so eingestellt werden, daß es ankommende Anrufe „beantwortet“ (Auto-Answer-Modus). Ein ankommender Anruf muß durch eine Loginprozedur zur Benutzeridentifikation beantwortet werden, die schließlich eine Shell mit den entsprechenden Rechten startet.

Zu diesem Zweck wird normalerweise von `init` ein `getty`-Prozeß für den seriellen Port gestartet.

## Das richtige getty

Für Linux gibt es mindestens vier verschiedene `getty`-Programme.

### **agetty**

von Wietse Venema aus Peter Orbaeks Utility-Sammlung. Das ist das einfachste `getty` für Linux. Es erfüllt seine Aufgabe - auf „seinem“ Port einen `login:` Prompt auszugeben und das `login`-Programm zu starten, sobald etwas auf diesem Port empfangen wird - zuverlässig, nicht mehr und nicht weniger.

### **getty\_ps**

von Paul Sutcliffe (ps) in der Linux-Version von Kris Gleason. Dieses `getty` benutzt eine `gettydefs` Datei wie das `getty` von System V. Ein zusätzliches `ugetty` ist speziell für die gemeinsame Benutzung einer seriellen Leitung für ein- und ausgehende Verbindungen vorbereitet. Die über Konfigurationsdateien und Kommandozeilenoptionen realisierbaren Variationen sind vielfältig und decken praktisch alle denkbaren Einsatzbedingungen ab. Weil das natürlich höhere Anforderungen an die Systemverwalterin stellt, kann das sowohl als Vor- als auch als Nachteil gewertet werden.

### **mgetty**

von Gert Doering. Neben der direkten Unterstützung ein- und ausgehender Verbindungen bietet

dieses `getty` die Möglichkeit, ankommende FAX-Sendungen zu bearbeiten. `mgetty` ist klein und einfach; es werden allerdings alle wesentlichen Einstellungen in die Binärdatei eincompiliert. Das bedeutet in der Regel, daß das `mgetty` speziell für das lokale System extra übersetzt werden muß.

## **modgetty**

von Walter Pelissero in der Linux-Version von Olaf Titz (hier nur der Vollständigkeit halber erwähnt). Dieses kann mit bestimmten Modems auch als Anrufbeantworter arbeiten, sowie FAXe empfangen. Es ist über eine eigene Scriptsprache programmierbar, dadurch sehr flexibel aber etwas komplizierter in der Handhabung.

## **Device-Handling**

Ein wichtiger Unterschied zwischen den `getty`-Versionen kommt unter dem Blickwinkel des Devicehandling zum Vorschein.

Das grundsätzliche Problem besteht darin, daß einerseits die serielle Modemleitung ständig vom `getty` überwacht werden soll, um eingehende Modemverbindungen anzunehmen, andererseits soll mit dem gleichen Modem eine ausgehende Verbindung aufgebaut werden können.

Die seriellen Schnittstellen werden durch die Gerätedateien `/dev/ttyS0`, `/dev/ttyS1`, `/dev/ttyS2` usw. im Filesystem dargestellt. Diese „normalen“ Devices blockieren ein Programm, das versucht, eine dieser Dateien zu öffnen, so lange, bis das Modem auf der „Carrier-Detect“-Leitung eine bestehende Verbindung signalisiert. Durch einen speziellen Parameter beim Aufruf des `open(2)` Systemcall kann diese Blockierung auch umgangen werden.

Wenn beispielsweise der UUCP-Dämon eine Modemverbindung aufbauen will und dazu das Modem über eine Gerätedatei anspricht auf der bereits ein `getty` wartet, wird die Verbindung zwar aufgebaut aber nach wenigen Sekunden wieder unterbrochen. Das dem Modemport wartendes `getty` kann nicht zwischen den Zeichen unterscheiden, die einerseits vom Modem an den Rechner, andererseits vom UUCP-Dämon an das Modem geschickt werden. Deshalb stört das `getty` die UUCP-Verbindung durch einen Loginprompt was schließlich zu einer Unterbrechung der Verbindung führt. Um das `getty` daran zu hindern, kann es nicht einfach durch ein `kill()`-Signal ausgeschaltet werden. In diesem Fall würde `init` sofort ein neues `getty` starten. Also müßte die Datei `/etc/inittab` entsprechend verändert und das `init` mit `kill -1` von dieser Änderung informiert werden, definitiv kein gangbarer Weg.

Die seit Kernel-Version 0.99.5 zusätzlich zu den `ttys*`-Devices vorhandenen Gerätedateien `/dev/cua0`, `/dev/cua1` usw. bilden die gleichen Schnittstellen nochmal ab. Beim Öffnen dieser Dateien wird ein Programm niemals blockiert. Allein wenn die parallele `/dev/ttyS?` Gerätedatei bereits geöffnet ist, wird vom `open(2)` Systemcall der Fehler `EBUSY` zurückgeliefert. Umgekehrt wird jeder Versuch, ein `/dev/ttyS?` zu öffnen, mit der gleichen Fehlermeldung abgewiesen, sobald das entsprechende `cua?`-Device erfolgreich geöffnet wurde. Durch diesen Mechanismus wird auf Kernelebene die gemeinsame Verwendung einer Modemleitung zum Ein- und Auswählen unterstützt.

Indem das `getty` beispielsweise auf `/dev/ttyS1` gestartet wird, kann der UUCP-Dämon auf dem parallelen Device `/dev/cua1` problemlos arbeiten. Solange nämlich vom Modem kein Carrier-Detect kommt, bleibt das `getty` blockiert. Wenn die Verbindung aufgebaut ist, das Carrier-Detect Signal also anliegt, ist die `/dev/cua1` Datei bereits durch `uucico` geöffnet, also bleibt das `getty` weiterhin blockiert.

Diese Methode stellt keine besonderen Ansprüche an das `getty`, ist also besonders zusammen mit

dem `agetty` zu empfehlen. Die modernen Modems können so konfiguriert werden, daß sie automatisch auf einen Wechsel des DTR-Signals vom Rechner einen Reset ausführen. Außerdem kann in einem nichtflüchtigen Speicher mindestens eine Reset-Konfiguration gespeichert werden. Auf diese Weise kann das Modem vollautomatisch ankommende Verbindungen aufnehmen und nach Beendigung irgendeiner Modemverbindung wieder in einen definierten Zustand zurückkehren. Gegen diese Methode ist einzuwenden, daß das Modem auch dann Anrufe entgegennimmt, wenn der Rechner - aus welchen Gründen auch immer - nicht dazu bereit ist.

Sowohl `getty_ps` als auch `mgetty` erlauben auch ohne (man kann sagen ``unter Umgehung der'') Kernelunterstützung die gleichzeitige Benutzung eines Modemports in beiden Richtungen. Dazu muß das Modem nicht im Auto-Answer-Modus sein. Um das zu erreichen, sind beide Programme darauf angewiesen, daß die um einen Port konkurrierenden Prozesse sogenannte Lockfiles anlegen. Dieses Verfahren wird - wenn auch auf freiwilliger Basis und in unterschiedlicher Form - von allen Programmen zur Datenfernübertragung eingehalten.

## Lockfiles

Die erste Voraussetzung zum Funktionieren dieses Systems ist die Übereinkunft, **wo** diese Lockfiles angelegt werden. Ein nach dem neuen File-System-Standard üblicher Ort ist das Verzeichnis `/var/spool/uucp`, es kann aber durchaus auch das „traditionelle“ Verzeichnis `/usr/spool/uucp` oder `/var/locks` sein. Die Lockfiles heißen dann beispielsweise `/var/spool/uucp/LCK..cua1` oder `/var/spool/uucp/LCK..ttyS0`. Die Namensendung entspricht also dem seriellen Port, der durch diese Datei „abgeschlossen“ werden soll.

Wenn zwei Programme dasselbe Verzeichnis für ihre Lockfiles benutzen, können sie vor dem Öffnen der Gerätedatei die Existenz eines solchen Lockfiles prüfen und gegebenenfalls auf den konkurrierenden Zugriff verzichten.

Weil diese Methode scheitert, wenn beispielsweise nach einem Programmabsturz so ein Lockfile erhalten bleibt, wird zusätzlich in dem Lockfile festgehalten, welcher Prozeß es angelegt hat. Allerdings gibt es auch hier wieder in der Form, **wie** diese Information abgelegt wird, verschiedene Varianten. Einige Programme hinterlassen die Prozeßnummer als Zahl aus ASCII-Ziffern, andere als 4-Byte-Integer im Binärformat. Wenn ein Programm auf einen seriellen Port zugreifen will und ein Lockfile für diesen Port findet, stellt es fest, ob der Prozeß, der dieses Lockfile angelegt, hat noch existiert. Ist das nicht der Fall, wird der Port trotz Lockfile geöffnet.

Weil unter Linux die Quelltexte zu allen in Frage kommenden Programmen vorhanden sind, ist die Wahrscheinlichkeit, daß sich alle Programme einer Distribution über die Verwendung der Lockfiles einig sind, größer, als es auf den ersten Blick den Anschein hat. Selbst wenn es im Einzelfall zu Problemen kommt, lassen die sich durch Neuübersetzen mit anderer Konfiguration verhältnismäßig leicht beheben.

Ein zusätzliches Problem entsteht, wenn zur „Vereinfachung“ ein Link auf die Schnittstellendatei angelegt wird. Zugegeben, `/dev/modem` ist aussagekräftiger als `/dev/cua1`. Das Lockfile `LCK..modem` kann aber keinesfalls verhindern, daß ein anderes Programm `/dev/cua1` öffnet.

Die dargestellten Probleme hätten längst zum völligen Aussterben von `getty_ps` und `mgetty` geführt, hätten sie nicht einige sehr überzeugende Features.

Der große Vorteil von `mgetty` liegt in seiner Fähigkeit, automatisch FAXe der Klasse 2 zu

empfangen, sofern das Modem dieses Protokoll beherrscht. Mit dem zusätzlichen `sendfax`-Programm ist damit der Linux-PC eine vollständige FAX-Station. Die Konfiguration von `mgetty` wird in die Binärdatei eincompiliert. Weil Gert Döring seinem „`mgetty+sendfax`“ Paket ein ausgezeichnetes Manual (englisch) beigelegt hat, wird hier auf eine weitere Beschreibung verzichtet.

## getty\_ps

Das `getty` von Paul Sutcliffe ist das flexibelste der hier vorgestellten Programme. Einige interessante Features sind:

- Logins können auf bestimmte Zeiten (Tage und Stunden) beschränkt werden.
- Durch „Klingelzeichen“ ist es möglich, auf einer Telefonleitung ankommende Daten- und Voiceverbindungen zu unterscheiden.
- Es können zwei Gerätedateien überwacht werden. Nach der Benutzung eines dieser Devices durch irgendein Programm wird das Modem neu initialisiert.
- Für jeden Port kann eine eigene Initialisierungsdatei eingesetzt werden.
- Die Parameter des Gerätetreibers lassen sich uneingeschränkt variieren.
- Das Modem muß nicht im Auto-Answer-Modus betrieben werden.

## Syntax:

```
getty [-R] [-C Connect] [-D Level] [-d Defaults-Datei] [-a] [-h] [-r Verzögerung] [-t Timeout]  
[-w Waitfor] Device [Label [Typ [LineD]]]
```

## Optionen:

**-R**

(RINGBACK) wie `RINGBACK=TRUE` im Defaults-File (Erklärung später)

**-C *Connect***

wie `CONNECT` im Defaults-File

**-D *Level***

setzt den Debuglevel, wie `DEBUG` im Defaults-File

**-a *Altline***

NICHT verwenden **\*\*BUG\*\***

**-c *gettydefs***

kann benutzt werden, um die *gettydefs*-Datei zu testen

**-d *Defaults-Datei***

veranlaßt `getty`, die angegebene *Defaults-Datei* anstelle der Datei `/etc/default/getty.Port` zu verwenden

**-h**

die gleiche Funktion wie `HANGUP=NO` im Defaults-File

**-r *Verzögerung***

die gleiche Funktion wie `DELAY=Verzögerung` im Defaults-File; stellt eine Zeitverzögerung zwischen dem Empfang des `WAITCHAR`-Zeichens und dem Erscheinen des Login-Prompts ein

### **-t *Timeout***

die gleiche Funktion wie `TIMEOUT= Timeout` im Defaults-File; `getty` wartet die angegebene Anzahl Sekunden nach der Ausgabe des Prompts auf einen Loginnamen, danach bricht es ab

### **-w *Waitfor***

das gleiche wie `WAITFOR=Waitfor` im Defaults-File; nach der Initialisierung des Data-Sets wartet `getty` auf die Zeichenkette *Waitfor* und gibt nach der DELAY-Verzögerung den Prompt aus

Nach den Optionen benötigt `getty` mindestens ein Kommandozeilenargument. Wenn weitere Argumente angegeben sind, werden sie in der Reihenfolge ihres Auftretens wie folgt interpretiert:

### ***Device***

(line) die Schnittstelle, auf der `getty` arbeiten soll; `getty` erweitert den Namen zu `/dev/Device`

### ***Label***

die Marke des Eintrags in `/etc/gettydefs`, mit dem `getty` die Schnittstelle initialisiert; wenn kein Label angegeben ist, wird der erste Eintrag in `gettydefs` genommen

### ***Typ***

ein gültiger `termcap`-Eintrag für das (erwartete) Terminal; der Eintrag wird benutzt, um eine Steuersequenz zum Bildschirmlöschen zu senden, außerdem wird der Typ in die `TERM`-Umgebungsvariable geschrieben

### ***LineD.***

die Line-Discipline (beispielsweise `SLIP`)

Seine enorme Flexibilität erhält das `getty_ps` durch zwei Initialisierungsdateien. Die eine, „Defaults-Datei“, enthält vor allem Einstellungen und „Chat-Scripts“ zur Koordination der Zusammenarbeit mit einem Modem oder einem Terminal, beispielsweise den String zur Geräteinitialisierung (Reset). Die andere, `/etc/gettydefs`, enthält die Daten zur Einstellung der seriellen Schnittstelle, also die Baudrate und ähnliches.

## **Defaults-Dateien**

`getty_ps` bietet die Möglichkeit, für verschiedene Devices unterschiedliche Konfigurationsdateien zu benutzen. Diese „Defaults-Dateien“ werden alle im Verzeichnis `/etc/default` erwartet. Die Namen der Konfigurationsdateien beginnen mit dem Programmnamen (`getty` oder `ugetty`) und tragen optional den Namen der Gerätedatei als Endung. Beispielsweise wird die Datei `/etc/default/getty` als Konfigurationsdatei für alle `getty_ps` auf allen Ports genommen, für die keine besondere Konfigurationsdatei existiert, `/etc/default/ugetty.ttyS1` ist die Konfigurationsdatei für ein `ugetty` auf dem Port `/dev/ttyS1`.

### **SYSTEM=*Name***

ist für den Systemnamen vorgesehen. Der angegebene Name wird durch das Symbol `@S` in der `ISSUE`-Meldung angezeigt. Wenn hier nichts angegeben wird, benutzt `getty` den `uname ( 2 )`-Systemaufruf, um den Nodenamen zu bestimmen.

### **VERSION=*Version***

setzt die Zeichenkette, die durch das Symbol `@V` in der `ISSUE`-Meldung ausgegeben wird.

Wenn die Zeichenkette *Version* mit einem Slash beginnt, wird sie als Pfadname einer Datei interpretiert (z. B. `/proc/version`).

### LOGIN=***Kommando***

bestimmt ein Kommando als Ersatz für `/bin/login`. Dieses Kommando wird dann mit dem Benutzernamen als einzigem Argument aufgerufen, um die Benutzeridentifikation durchzuführen.

### INIT=***ChatString***

bestimmt einen Chat-String zur Initialisierung des seriellen Gerätes. Ein Chat-String besteht aus einer Kette von expect-send-Paaren. Jedes Paar besteht aus zwei durch Leerzeichen getrennten Zeichenketten. Das erste Auftreten eines expect-Strings im Eingabedatenstrom führt zur Ausgabe des send-Strings auf die serielle Schnittstelle. Im Unterschied zur UUCP-Konfiguration werden die send-Strings nicht automatisch durch ein Newline abgeschlossen.

Ein expect-String kann durch 'expect-send-expect' Schleifen unterbrochen werden. Diese send-Einschübe werden gesendet, wenn innerhalb einer kurzen Frist das erwartete expect-Wort nicht angekommen ist (typisches Beispiel aus einem anderen Bereich ist die „`ogin:-BREAK-ogin:``` Sequenz, mit der `uucico` so lange BREAK sendet, bis ein Login: Prompt empfangen wird)

Die folgenden Sonderzeichen können im INIT-String verwendet werden:

`\\`

der Backslash selbst

`\b`

Backspace (^H)

`\c`

unterdrückt den Zeilenvorschub am Ende einer Zeichenkette

`\f`

Seitenvorschub (^L)

`\n`

Zeilenvorschub (^J)

`\r`

Wagenrücklauf (^M)

`\s`

Leerzeichen

`\t`

horizontaler Tabulator

`\nnn`

ASCII Zeichen mit der (dezimalen) Nummer *nnn*; oktale und hexadezimale Darstellungen sind mit den üblichen Präfixen erlaubt

`\p`

eine Sekunde Pause

`\d`

zwei Sekunden Pause

\K

1/4 Sekunde Break

\T*nnn*

Verändert das allgemeine Timeout auf die in *nnn* angegebene Anzahl Sekunden; der Wert kann wieder dezimal, oktal oder hexadezimal angegeben werden

INITLINE=*Device*

Wenn uugetty auf einem blockierenden ttyS? Device arbeitet, kann durch Angabe einer INITLINE das parallele cua? Device für die Initialisierung und gegebenenfalls für die WAITCHAR/WAITFOR-Sequenz benutzt werden. Das uugetty vermeidet durch Überwachung der Lockfiles für beide Devices konkurrierende Zugriffe und initialisiert nach jeder Benutzung das Gerät neu. In einigen Versionen von getty\_ps ist die gleiche Funktion unter dem Namen ALTLINE implementiert.

ISSUE=*Meldung*|*Datei*

Vor dem Loginprompt gibt getty normalerweise eine kurze Meldung aus, die beispielsweise den Systemnamen beinhaltet. Traditionell wird diese Meldung aus der Datei `/etc/issue` gelesen. getty\_ps ermöglicht es, eine andere Datei zu benennen, aus der die Meldung gelesen werden soll, oder die Meldung direkt anzugeben.

In der Meldung werden bestimmte Sonderzeichen erkannt und ersetzt:

\\

der Backslash selbst

\b

Backspace (^H)

\c

unterdrückt den Zeilenvorschub am Ende einer Zeichenkette

\f

Seitenvorschub (^L)

\n

Zeilenvorschub (^J)

\r

Wagenrücklauf (^M)

\s

Leerzeichen

\t

horizontaler Tabulator

\i*nnn*

ASCII Zeichen mit der (dezimalen) Nummer *nnn*; oktale und hexadezimale Darstellungen sind mit den üblichen Präfixen erlaubt

Ein einzelner Backslash am Zeilenende führt dazu, daß die Ausgabe in derselben Bildschirmzeile fortgesetzt wird.

Die folgenden Parameter werden erkannt und ersetzt:

@B

die aktuelle Baudrate

@D

das aktuelle Datum im Format MM/DD/YY

@L

der Port, mit dem `getty` verbunden ist

@S

der Systemname (siehe oben)

@T

die aktuelle Zeit im Format HH:MM:SS

@U

die Anzahl der aktuell eingeloggten User; diese Zahl wird aus den Einträgen in der `utmp`-Datei errechnet

@V

die Zeichenkette aus `VERSION` (siehe oben)

@@

der Klammeraffe selbst

**CLEAR=***YES*

Solange der Schalter mit `YES` belegt wird, versucht `getty_ps`, den Bildschirm vor der Ausgabe der `ISSUE` Meldung zu löschen.

**HANGUP=***YES*

Solange die Einstellung `HANGUP=YES` besteht, wird `getty` veranlaßt, eine bereits vor dem Programmstart bestehende Verbindung durch ein `HANGUP`-Signal zu unterbrechen.

**WAITCHAR=***NO*

Mit `WAITCHAR=YES` wird `getty` veranlaßt, auf ein einzelnes Zeichen zu warten und erst dann mit der Initialisierung fortzufahren. Auf diese Weise kann beim Betrieb eines Terminals mit dauernd anliegendem Carrier-Detect-Signal eine Endlosschleife unterbrochen werden.

**DELAY=***Sekunden*

Zusammen mit `WAITCHAR` wird hier eingestellt, wie lange auf das Zeichen gewartet werden soll. Der voreingestellte Wert 0 veranlaßt `getty_ps`, beliebig lange auf ein Zeichen zu warten.

**TIMEOUT=***Sekunden*

Hier kann eine Zeitspanne angegeben werden, die von `getty_ps` nach der Ausgabe des Prompts auf einen Benutzernamen gewartet wird. Wenn nichts anderes bestimmt wird, wartet `getty` endlos.

**WAITFOR=***Wort*

arbeitet wie `WAITCHAR`, nur daß auf die Zeichenkette *Wort* gewartet wird.

**CONNECT=***ChatString*

der *Chat-String* ist eine expect/send-Sequenz, mit der `getty` den Aufbau einer ankommenden Modemverbindung regeln kann. Ein typisches Beispiel ist:



WAITFOR=RING

WAITFOR=RING

CONNECT= " " ATA\r CONNECT\s\A

Zuerst wird hier mit WAITFOR zweimal auf die Zeichenkette „RING“ gewartet. Danach wird in dem ChatScript auf nichts weiter gewartet ( " " ), also sofort ein ATA zum Modem geschickt um es zum Abheben zu veranlassen. Dann wird auf eine Zeichenkette der Form „CONNECT 14400“ gewartet. Die Sonderzeichen haben folgende Bedeutung:

\A

die Baudrate

\s

ein Leerzeichen

ALTLOCK=**Port**

diese Variable wird nur von uugetty benutzt. Der angegebene Port wird wie der „eigene“ überwacht und durch ein Lockfile blockiert, sobald eine Verbindung erkannt wird. Wenn ein anderes Programm einen der überwachten Ports benutzt (durch ein Lockfile blockiert), wird nach der Freigabe das Gerät neu initialisiert.

ALTLINE=**Port**

siehe INITLINE

RINGBACK=**NO**

Wenn RINGBACK=YES gesetzt ist, wird der oben beschriebene WAITFOR . . . CONNECT Mechanismus unterbrochen, und ankommende Anrufe werden zuerst abgewiesen. Wenn nach ein- bis dreimaligem Klingeln eine Pause von höchstens 60 Sekunden eingelegt wird, nimmt das Modem nach dem nächsten Klingeln ab. Durch dieses „Klingelzeichen“ kann eine Telefonleitung gleichzeitig für Gespräche und Datenverbindungen genutzt werden.

MINRBTIME=**Sekunden**

minimale Verzögerung zum zweiten Anlauf (Voreinstellung 6 Sek.)

MAXRBTIME=**Sekunden**

maximale Verzögerung zum zweiten Anlauf (Voreinstellung 60 Sek.)

INTERRING=**Sekunden**

die maximale Zeit zwischen zwei Klingelphasen (Voreinstellung 4 Sek.)

MINRINGS=**Anzahl**

minimale Anzahl der Klingelphasen bei der ersten Anwahl (Voreinstellung 1)

MAXRINGS=**Anzahl**

maximale Anzahl der Klingelphasen bei der ersten Anwahl (Voreinstellung 3)

SCHED=**Zeitraum1 [Zeitraum2]**

In der Variablen SCHED können Zeiträume bestimmt werden, in denen getty\_ps aktiv ist und so ein Login auf der seriellen Leitung ermöglicht. Ein Zeitraum wird in der Form Tag:Std:Min-Tag:Std:Min eingegeben. Tag ist eine Zahl zwischen 0 (Sonntag) und 6 (Samstag). Wenn die aktuelle Zeit in einem der Zeiträume liegt, wird das Modem initialisiert und bis zum Ende des Zeitraums auf ein Login gewartet. Am Ende jedes Zeitraums wird der OFF ChatString zum Modem geschickt.

OFF=**ChatString**

entspricht dem `INIT` ChatString, wird aber zum Deaktivieren des Modems nach einem Scheduler Zeitraum benutzt.

## **DEBUG=Level**

setzt den Debuglevel. Der Level wird als Oktalzahl angegeben, die Bits schalten jeweils eine Ebene hinzu:

### **OPT**

(001) Interpretation der Kommandozeile (`getopt(3)`)

### **DEF**

(002) Bearbeitung des Defaults-File

### **UTMP**

(004) Update der `utmp/wtmp`-Einträge

### **INIT**

(010) Geräteinitialisierung (`INIT`)

### **GTAB**

(020) Bearbeitung der `gettytab`-Datei

### **GETL**

(040) Loginname einlesen

### **RUN**

(100) andere Laufzeitfehler

### **LOCK**

(200) Lockfile-Handling (nur `uugetty`)

### **SCHED**

(400) Scheduling

Um alle Debugmeldungen zu erhalten, muß Level 0777 eingestellt werden.

## **gettydefs**

Die Datei `/etc/gettydefs` wird vom `getty_ps` ausgewertet, das daraus die Parameter zur Einstellung des Terminaldevices bezieht.

Jeder Eintrag besteht aus einer Zeile der Form:

Label# Startparameter # Loginparameter #Prompt# nächstes Label

Ein Hashmark '#' am Zeilenanfang leitet einen Kommentar ein.

Die Felder eines Eintrags haben folgende Bedeutung:

### **Label**

gibt der Zeile einen „Namen“. Damit kann dem `getty_ps` auf der Kommandozeile der Starteintrag übergeben werden.

### **Startparameter**

ist eine durch Leerzeichen getrennte Liste von symbolischen Konstanten zur Einstellung des Gerätetreibers nach dem ersten Öffnen durch `getty_ps`. Es können alle Parameter benutzt werden, die vom `ioctl(2)`-Systemaufruf verstanden werden. Die erlaubten Symbole sind in

/usr/include/linux/termios.h definiert. Erklärt sind sie beim [stty-Kommando](#).

Die Angabe einer Leitungsgeschwindigkeit ist notwendig, alle anderen Parameter optional.

## Loginparameter

ist eine durch Leerzeichen getrennte Liste von symbolischen Konstanten zur Einstellung des Gerätetreibers vor dem Aufruf des login-Kommandos durch `getty_ps`. Da `getty_ps` durch diesen Aufruf verdrängt wird, sind das die Einstellungen, die beim Start der ersten Shell aktiv sind.

Für die erlaubten und notwendigen Konstanten gilt das gleiche wie für die Startparameter.

## Prompt

ist der Loginprompt von `getty_ps`. Leerzeichen und `TAB` werden so ausgegeben, wie sie in diesem Feld erscheinen. Außerdem sind alle Symbole und Sonderzeichen erlaubt, die auch in der ISSUE-Meldung benutzt werden dürfen.

## nächstes Label

zeigt auf ein definiertes Label (erstes Feld) einer Zeile in `/etc/gettydefs`. Der durch dieses Label bezeichnete Eintrag wird angesprungen, wenn anstelle eines Buchstabens ein `BREAK` empfangen wird. Wenn ein Modem auf einer zu hohen Leitungsgeschwindigkeit wartet, wird ein `RETURN` wie ein `BREAK` interpretiert.

Durch Verkettung mehrerer Einträge mit (absteigend) unterschiedlichen Baudraten kann ein langsamer ankommender Anruf das `getty_ps` dazu bringen, das lokale Modem so lange schrittweise zu verlangsamen, bis eine Verständigung möglich ist.

Die modernen Modems mit Datenkompression handeln mit ihrer Gegenstelle automatisch die höchste mögliche Geschwindigkeit aus und werden deshalb zur Computerseite mit einer festen Baudrate betrieben, die höher sein sollte als jede real auf der Telefonleitung auftretende. In diesem Fall zeigt der Eintrag für das nächste Label auf das der aktuellen Zeile.

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [UUCP - Das Internet der](#) **Up:** [Datenreisen und reisende Daten](#) **Previous:** [Kontaktaufnahme](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

**Next:** [Elektronische Post mit smail](#) **Up:** [Datenreisen und reisende Daten](#) **Previous:** [Seriell einloggen](#)

## Subsections

- [UUCP-Anschluß - aber wie?](#)
  - [Was kann UUCP?](#)
  - [Wie sicher ist UUCP?](#)
  - [Taylor-UUCP](#)
  - [Überblick über die Konfigurationsdateien](#)
  - [Log-Dateien](#)
  - [Die config-Datei](#)
  - [Die sys-Datei](#)
    - [Telefonnummer und Login-Information](#)
    - [Erlaubte Zeiten](#)
    - [Das Chat-Skript](#)
    - [Protokoll-Parameter](#)
  - [Die port-Datei](#)
  - [Die dial-Datei](#)
  - [Testen der Konfiguration](#)
  - [Regelmäßige Verbindungen](#)
- 

# UUCP - Das Internet der Armen Leute

Wer seinen Rechner an ein Netzwerk anschließen will und nicht gerade das Glück hat, über einen lose herumliegenden Internet-Anschluß zu stolpern, muß sich gewöhnlich nach kostengünstigeren Alternativen umschaun. Die erste und beste Wahl ist in diesem Fall das Telefon. In Deutschland existieren mehrere, zum Teil bundesweite Computernetze, die ihre Mitglieder auf diese Weise miteinander verbinden.

Ein Teil der Netze, vor allem diejenigen, deren Mehrzahl aus UN\*X-Systemen besteht, verwenden UUCP, um über das Telefonnetz miteinander zu kommunizieren. Der Name UUCP ist eine Abkürzung und steht für Unix-to-Unix-Copy. UUCP wurde 1978 von den Bell Laboratories entwickelt, um einen flexiblen Datenaustausch zwischen einzelnen Zweigniederlassungen zu ermöglichen. Dank seines einfachen Designs und seiner relativen anspruchslosigkeit, was Installation und Wartung betrifft, hat sich UUCP rasch einen Spitzenplatz erobert, den es wohl noch einige Jahre behalten wird.

Im Laufe der Jahre hat es verschiedene Implementationen von UUCP gegeben. Dabei können grob zwei "Grundtypen" unterschieden werden, die sich im Format der Konfigurationsdateien stark unterscheiden. Der eine Typ, sogenannte "Version 2"-Implementationen, folgen einem einfacheren Konzept und stellen den älteren Zweig der Familie dar (Version 1 wurde nur innerhalb der Bell Laboratories verwendet). Der neuere Zweig stammt von einer Implementation aus 1983 von P. Honeyman, D.A. Novitz und B.E. Redman ab und heißt nach seinen Autoren auch HoneyDanBer oder HDB. Das HoneyDanBer-UUCP wurde unter der Bezeichnung *Basic Network Utilities* (BNU) ins AT&T Unix SVR3 integriert, so daß dieser Name ebenfalls in Verwendung ist.


# UUCP-Anschluß - aber wie?

Ein zentrales Dogma im Usenet, dem weltweiten News-Netz, ist ``*Get a feed and you're on it,*'' zu deutsch: Besorg' Dir einen Anschluß, und Du bist dabei. Das ist oft leichter gesagt als getan, denn Informationen über Anschlußmöglichkeiten erhält man oft nur, wenn man bereits Kontakt zu dieser Netzwelt aufgenommen hat.

Derzeit gibt es zwei größere gemeinnützige Vereine in Deutschland, die ihren Mitgliedern Netz-Dienste wie Mail und News über UUCP - und in zunehmendem Maße auch über TCP/IP - bieten. Dies sind einerseits *Individual Network e.V.* (kurz IN), andererseits der *Verein zur Förderung der privat betriebenen Datenkommunikation e.V.* (kurz Vz\* oder auch Sub-Netz, weil sich den vollen Namen im allgemeinen niemand merken kann). Beide Vereine kaufen Kontingente bei kommerziellen Service-Providern ein (s.u.), um auch die Versorgung mit internationaler Mail und News bieten zu können. Gleichzeitig gibt es immer mehr Domains im IN und VzFdpbDk, die über ISDN ans Internet angeschlossen sind, so daß dort auch Dienste wie FTP, WWW und anderes angeboten werden. Während der VzFdpbDk in gewissem Rahmen auch kommerziellen Kunden offensteht, können dem IN nur Privatpersonen beitreten. Kontaktadressen für beide Vereine entnehmen Sie bitte dem Anhang.

Neben diesen gemeinnützigen Vereinen existiert die Möglichkeit, die Dienste von kommerziellen Providern in Anspruch zu nehmen. Mittlerweile bieten viele Provider Mail und News auch für private Benutzer an, liegen aber mit ihren Preisen im Allgemeinen immer noch wesentlich über denen des IN und des VzFdpbDk.

## Was kann UUCP?

UUCP ermöglicht es, das öffentliche Telefonnetz zur Datenübertragung zu verwenden. Im Unterschied zu Terminal-Programmen, wie sie im vorigen Abschnitt vorgestellt wurden, arbeitet es aber nicht interaktiv, sondern erlaubt Ihnen, über verschiedene Dienstprogramme Aufträge zu erteilen, die zu festgelegten Zeiten selbsttätig ausgeführt werden. Um beispielsweise eine Datei von einem fremden Rechner anzufordern, können Sie folgendes Kommando ausführen: 

```
$ uucp -r flarp\!\~/archiv/Inhalt.Z Inhalt.Z
```

Dies erteilt UUCP den Auftrag, bei der nächsten Verbindung, die es mit dem System *flarp* aufbaut, aus einem allgemein zugänglichen Verzeichnis (unter Linux ist das nach dem neuen File-System-Standard `/var/spool/uucppublic`) die Datei `archiv/Inhalt.Z` zu übertragen und im momentanen Verzeichnis in `Inhalt.Z` abzulegen. Das Flag `-r` fordert *uucp* auf, nicht sofort zu versuchen, die Verbindung aufzubauen.

Ebenso können Sie mit Hilfe von UUCP Kommandos auf einem fremden Rechner ausführen - sofern dieser es Ihnen erlaubt. Um beispielsweise die Datei `handbuch.dvi` auf dem Rechner *flarp* auszudrucken, müßten Sie das folgende eingeben:

```
$ uux -r flarp\!lpr -d \!handbuch.dvi
```

Das Ausrufezeichen vor `handbuch.dvi` weist *uux* darauf hin, daß es sich dabei nicht um ein einfaches Befehlsargument handelt, sondern um eine Eingabedatei, die zusätzlich zu dem Druckauftrag übertragen werden muß.

Wie Sie an den Beispielen sehen, verwendet UUCP das Ausrufezeichen, um den Namen eines Rechners von weiteren Parametern wie Datei- oder Befehlsnamen zu trennen. Es kann aber auch dazu verwendet werden, mehrere Rechnernamen voneinander zu trennen. Dieser Fall tritt beispielsweise auf, wenn Sie auf ein System zugreifen wollen, das Sie nicht direkt erreichen können. In diesem Fall müssen Sie auf die Vermittlung eines

Systems zurückgreifen, das bereit ist, Ihren Auftrag an den Zielrechner weiterzuvermitteln. Läge in obigem Beispiel das Archiv nicht auf *flarp*, sondern auf *tauris*, einem Nachbarn von *flarp*, so müßte der Befehl stattdessen lauten:

```
uucp -r flarp\!tauris\!\~/archiv/Inhalt.Z Inhalt.Z
```

UUCP wird in diesem Beispiel den Auftrag, die Datei von *tauris* zu kopieren, an *flarp* weiterreichen. Zu geeigneter Zeit wird *flarp* diesen dann ausführen und die Datei bei sich zwischenspeichern. Bei der nächsten Verbindung mit Ihrem System wird diese dann endgültig an Sie übertragen.

Wie Sie sehen, gibt *flarp!tauris* hier den Pfad von Ihrem System zum Zielsystem an. Da das Ausrufezeichen im Computer-Englisch oft als *bang* bezeichnet wird, bezeichnet man dies auch als *bang path*.

Neben diesen allgemeinen, allen Benutzern zugänglichen Befehlen kennt UUCP noch weitere, die hinter den Kulissen für das eigentliche Funktionieren eines UUCP-Knotens sorgen. Dies ist einerseits *uucico*, das Haupt- und Staats-Programm, das für die eigentliche Übermittlung der Daten zwischen zwei Systemen zuständig ist. Befehle wie *uucp* legen nämlich die zu übertragenden Daten und die Beschreibung ihres Auftrags im allgemeinen nur in einer Temporärdatei im sogenannten Spool-Verzeichnis ab.

Um beispielsweise den *uux*-Auftrag aus dem vorangehenden Beispiel an *flarp* zu übertragen, muß eine Verbindung zu diesem System hergestellt werden. Dies geschieht durch den Befehl

```
$ uucico -s flarp
```

*uucico* versucht nun, die (Telefon-) Verbindung mit *flarp* aufzubauen, und loggt sich dort ein. Dabei wird auf *flarp* ebenfalls ein *uucico* gestartet, mit dem *uucico* dann Daten austauscht. Im vorliegenden Falle sendet der lokale *uucico* die Datei *handbuch.dvi* an *flarp*, sowie den Auftrag, mit dieser Datei das Druckprogramm *lpr* aufzurufen.

Dieser Auftrag wird auf *flarp* anschließend von *uuxqt* ausgeführt, das nach Beendigung der Verbindung automatisch von *uucico* aufgerufen wird.

## Wie sicher ist UUCP?

Diese Frage hat sich Ihnen vielleicht sofort aufgedrängt, als in den vorigen Abschnitten so freizügig die Rede davon war, daß mit UUCP Kommandos auf anderen Rechnern ausgeführt und Dateien nach Belieben hin- und herkopiert werden konnten.

Dieser Eindruck ist etwas irreführend. All diese Sachen sind *im Prinzip* möglich; es hängt aber immer vom Verwalter des einzelnen Systems ab, was er einzelnen Gästen erlaubt, die über UUCP auf die Maschine zugreifen. Gewöhnlich erlaubt UUCP fremden Systemen nur, zwei Kommandos auszuführen: *rmail* und *rnews*, die traditionellen Befehle zum Einliefern von Mail und News. Ebenso ist es fremden Systemen in der Regel nicht erlaubt, einfach beliebige Kommandos auf Ihrem System ausführen zu lassen, wie es beispielsweise mit dem Druckbefehl im Beispiel oben geschah. Es liegt in Ihrem Belieben, ob Sie einigen Systemen weitergehende Rechte einräumen, oder diese sogar weiter einschränken.

Normalerweise bietet UUCP eine Art ``rechtsfreien Raum": in der Voreinstellung dürfen fremde Systeme im Verzeichnis */var/spool/uucppublic* beliebige Daten ablegen oder sie daraus herunterladen.


Im allgemeinen wird man diese Voreinstellungen unverändert bestehen lassen. Bitte lesen Sie in den Handbüchern oder weiterführender Dokumentation nach, wenn Sie diese ändern wollen.

# Taylor-UUCP

Die auf UN\*X-Systemen am weitesten verbreitete "freie" UUCP-Implementation ist *Taylor UUCP*, so genannt nach ihrem Autor, Ian Taylor. Die derzeit aktuelle Version ist Version 1.5, die auch in den meisten Linux-Distributionen und CDs enthalten ist.

Eine Besonderheit von Taylor-UUCP ist, daß es sowohl die Konfigurationsformate des Version 2-UUCP als auch die der HoneyDanBer-Implementationen versteht. Diese Formate lassen sich bei der Kompilierung voreinstellen. Daneben gibt es noch ein drittes Format, das sich vor den anderen durch eine wesentlich übersichtlichere Syntax auszeichnet. Im folgenden wird diese Taylor-spezifische Konfiguration behandelt. Die allermeisten der in den diversen Linux-Paketen enthaltenen UUCP-Binaries unterstützen dieses Konfigurationsformat.

Taylor-UUCP ist nicht nur in der Lage, Verbindungen über Telefonleitungen und Modems aufzubauen, sondern kann auch Transfers über ein TCP/IP-Netzwerk durchführen. Es würde hier allerdings zu weit führen, dies auch zu diskutieren, weshalb sich das hier beschriebene Beispiel auf den "traditionellen" Fall einer Wählverbindung beschränkt. Interessierte Leserinnen und Leser seien auf weiterführende Dokumentation verwiesen.

Besonders empfehlenswert ist in diesem Zusammenhang die umfangreiche Original-Dokumentation zu Taylor-UUCP, die im TEXinfo-Format vorliegt. Sie ist auf den meisten FTP-Sites  erhältlich, die GNU-Software anbieten. Einige Linux-Distributionen dürften sie auch enthalten. Ebenfalls zu empfehlen ist das bei O'Reilly & Associates erschienene Buch *Managing UUCP and Usenet*, das zwar auch (noch) nicht die Taylor-Konfiguration beschreibt, aber ansonsten alles behandelt, was im Zusammenhang mit UUCP von Interesse ist.

## Überblick über die Konfigurationsdateien

Bevor wir uns den einzelnen für die Konfiguration nötigen Schritten zuwenden, lohnt es sich, darüber nachzudenken, welche Informationen UUCP überhaupt benötigt, um mit einem anderen System zu kommunizieren und Daten auszutauschen.

Nehmen wir beispielsweise an, Sie wohnen in Berlin und haben sich bereits mit der Firma LunetIX in Verbindung gesetzt, die auf ihrer Maschine *cicero* ein Gateway für Mail und News betreibt. LunetIX hat Ihnen einen UUCP-Zugang eingerichtet. Als UUCP-Namen des Rechners haben Sie vorläufig *isis* vereinbart. Ihr Login-Name ist in diesem Fall *uisis*, als Paßwort ist *gniggl* eingetragen, und die Telefonnummer ist 123 45 67. Damit hätten Sie zunächst die Information über den Zielrechner zusammen.

Aber das reicht natürlich nicht. Um überhaupt anrufen zu können, muß UUCP natürlich auch wissen, an welcher seriellen Schnittstelle sich das Modem überhaupt befindet und wie es angesprochen wird. Wir nehmen an, daß das Modem an COM2 hängt und ein Hayes-kompatibles Modem mit einer maximalen Übertragungsrate von 9600bps ist.

Mit diesen Informationen sollte UUCP nun in der Lage sein, eine Verbindung mit *cicero* herzustellen. Bei systematischer Betrachtung lassen sich die Informationen in mehrere Sinneinheiten zusammenfassen:

- Zunächst gibt es *systemspezifische* Informationen, wie die Telefonnummer, Login und Paßwort. Zu diesen Informationen kann unter Umständen hinzukommen, daß Sie überhaupt nur zu bestimmten Zeiten eine Verbindung mit dem anderen System aufbauen dürfen, um andere Benutzer nicht zu behindern.
- Eine weitere Klasse von Informationen betrifft die Hardwareaspekte des angeschlossenen Übertragungsgerätes: an welcher Schnittstelle ist es zu finden, was ist seine maximale Übertragungsgeschwindigkeit, und von welchem Typ ist es?

- Die dritte Klasse von Informationen betrifft schließlich die logische Ansteuerung für Geräte eines bestimmten Typs: wie kann ein bestimmtes Gerät angesprochen werden, d.h. welche Befehle müssen ausgegeben werden, damit beispielsweise eine bestimmte Telefonnummer gewählt wird? Im allgemeinen wird es sich hierbei um modemspezifische Daten handeln; für TCP/IP-Links oder direkte serielle Leitungen beispielsweise können diese aber auch anderer Art sein.

Genau entlang dieser Einteilung ordnet Taylor-UUCP die benötigten Informationen nun auch drei verschiedenen Konfigurationsdateien zu: `sys` für die systemspezifischen Daten, `ports` für die Konfiguration der Schnittstellen und die Zuordnung der angeschlossenen Geräte, und schließlich `dial` für die Beschreibung der Modemtypen. Allgemeine Daten, wie beispielsweise der UUCP-Name Ihres Rechners, sind in der Datei `config` abgelegt.

Alle diese Dateien, die im folgenden einzeln beschrieben werden, haben ein gemeinsames Format. Ein Eintrag besteht immer aus einem Schlüsselwort, gefolgt von Leerzeichen und/oder Tabulatorzeichen sowie einem oder mehreren Argumenten. Er kann über das Zeilenende fortgesetzt werden, wenn das letzte Zeichen der Zeile ein Backslash (`\`) ist. Kommentare werden durch ein Doppelkreuz (`#`) eingeleitet und erstrecken sich ebenfalls bis zum Zeilenende.

Die Konfigurationsdateien sind in einem gemeinsamen Verzeichnis versammelt. Nach dem Filesystem Standard sollten sie in `/usr/lib/uucp` zu finden sein; es waren aber auch Verzeichnisse wie `/usr/local/lib/uucp` oder `/usr/conf/uucp` in Gebrauch. Die Slackware-Distribution schließlich legt die Dateien in Unterverzeichnissen von `/usr/lib/uucp` ab - wenn Sie das HDB-Format wählen, müssen Sie die Dateien in `hdb_config` einrichten; im Falle der Taylor-Konfiguration heißt das Verzeichnis `taylor_config`.

Wenn Sie UUCP aus einer der diversen Linux-Distributionen installieren, wird dieses Verzeichnis normalerweise angelegt und enthält oft auch Beispieldateien.

Wenn Sie Taylor-UUCP in guter alter Tradition sogar selbst kompilieren, können Sie dieses Verzeichnis auch selber festlegen. Sollten Sie Linux in einem lokalen Netz betreiben und das `/usr`-Verzeichnis verteilt nutzen (d.h. per NFS von einem zentralen Server mounten), können Sie dem File-System-Standard entsprechend die UUCP-Konfigurationsdateien in das Verzeichnis `/etc` oder `/etc/uucp` installieren.

Im folgenden wird angenommen, daß die UUCP-Programme so konfiguriert wurden, daß sie die Konfigurationsdateien an ihrem traditionellen Platz im Verzeichnis `/usr/lib/uucp` suchen.


## Log-Dateien

Da UUCP-Befehle oft im Hintergrund aufgerufen werden, senden sie Ausgaben wie Fehler- oder Diagnose-Meldungen nicht auf die Konsole, sondern in mehrere Log-Dateien, die unterhalb des Verzeichnisses `/var/spool/uucp` abgelegt werden.

Abhängig davon, wie UUCP konfiguriert wurde, werden die Log-Files unterschiedlich gehandhabt. Die meisten UUCP-Pakete unter Linux sind dafür konfiguriert, HDB-kompatible Log-Dateien zu erzeugen. Diese Konvention kennt einerseits gewöhnliche Log-Dateien, in denen jede Transaktion vermerkt wird, und Debugging-Logs, in die zusätzliche Information abgelegt werden kann, wenn es vom Benutzer verlangt wird.

Die erste Kategorie von Log-Dateien ist unterhalb von `/var/spool/uucp/.Log` zu finden. Dieses Verzeichnis ist in drei weitere Unterverzeichnisse aufgeteilt; je eins für `uucp`, `uux` und `uucico`. Jedes dieser Unterverzeichnisse enthält ein separates Log für jedes fremde System. Ein Auftrag an `uux`, auf `flarp` das Handbuch auszudrucken, würde beispielsweise von `uux` in der Datei `uux/flarp` vermerkt. Beim nächsten Verbindungsaufbau mit `flarp` würde dann `uucico` einen weiteren Eintrag in der Datei `uucico/flarp` machen, mit dem die Übertragung des Auftrags protokolliert würde.



Die zweite Kategorie von Log-Dateien wird nur erzeugt, wenn Sie dies ausdrücklich von einem Programm verlangen. Debugging-Logs werden gewöhnlich nur benötigt, um Fehlern nachzugehen. Diese Daten werden ausgegeben, indem Sie als Kommando-Parameter die Option `-x` mit einem Argument zwischen 1 und 11 angeben. Das Argument gibt an, wie detailliert die Information sein soll; höhere Werte bedeuten mehr Information. Diese Informationen werden in der Datei `audit.local` im Verzeichnis `/var/spool/uucp/.Admin` abgelegt. Es ist auch möglich, daß ein fremdes UUCP-System zu Beginn einer Verbindung Ihr System veranlaßt, Debug-Informationen aufzuzeichnen; diese wird dann in der Datei `audit` anstelle von `audit.local` abgelegt. Dies können Sie mit Taylor-UUCP dadurch erreichen, daß Sie zusätzlich die Option `-X` mit einem entsprechenden Wert zwischen 1 und 11 angeben. 

Ebenfalls im Verzeichnis `.Admin` findet sich schließlich auch die Datei `xferstats`, in der die übertragenen Dateien mit Größe, Übertragungsdauer, usw aufgezeichnet werden. Sie wird von einigen im Quellcode zu Taylor-UUCP enthaltenen Utilities genutzt, um Benutzungsübersichten und ähnliches zu erstellen.

Ist Ihr UUCP so konfiguriert, daß er Taylor-spezifische Log-Dateien erzeugt, finden Sie diese in `/var/spool/uucp` als Log beziehungsweise Debug und Stats.

## Die config-Datei

Die `config`-Datei kann zentral die in den Binärdateien gespeicherten Einstellungen verändern. Beispielsweise können hier Namen und Anordnung der Konfigurationsdateien und deren Formate neu definiert werden. Allerdings sind alle möglichen Parameter mit sinnvollen Voreinstellungen belegt.

Der wichtigste und meist einzige Eintrag, den Sie in dieser Datei vornehmen müssen, ist der UUCP-Name Ihres Rechners:

```
# /usr/lib/uucp/config
hostname      isis
```

Mehr brauchen Sie hier zunächst nicht einzutragen.

Falls Sie Ihr System auch für Anonymous UUCP öffnen wollen, werden in diese Datei auch die Zugangsrechte für unbekannte UUCP-Systeme eingetragen. Für Details konsultieren Sie bitte die Original-Dokumentation zu Taylor-UUCP.

## Die sys-Datei

So einfach wie mit der `config`-Datei geht es natürlich nicht weiter. Bei den System-Daten wird es schon etwas kniffliger. Wie bereits erwähnt, werden diese in der Datei `sys` eingetragen. Das folgende Beispiel zeigt die passenden Einträge für die Verbindung zum Rechner `cicero`, wie sie am Anfang des Kapitels vorgestellt wurde.

```
# /usr/lib/uucp/sys
# Defaults
chat          ogin: \L ssword: \P
port          modem1
time          Any 2
protocol-parameter g window      7
protocol-parameter g packet-size 512

# 1. cicero.
```

```
# Kontakt: Sebastian; Tel: 1234568
system          cicero
phone           1234567
call-login      isis
call-password   gniggl

# 2. flarp
# Archiv-Site; Anonymous UUCP.
# Bem.: Miese Telefonleitung.
system          flarp
phone           7654321
call-login      uucp
call-password   nuucp
protocol-parameter g window          7
protocol-parameter g packet-size 64
```

Diese sys-Datei beschreibt zwei Systeme, *cicero* und *flarp*. Jede Systembeschreibung beginnt in einer Zeile, die das Schlüsselwort `system` enthält. Alle Informationen, die zwischen zwei `system`-Zeilen auftauchen, beziehen sich ausschließlich auf die angegebene Site.

Anweisungen, die *vor* der ersten `system`-Zeile auftreten, werden als Default-Angaben für alle Systeme verwendet. Taucht dieselbe Angabe auch noch im System-Eintrag auf, so verdeckt diese Angabe den Default für das spezielle System. Beispielsweise haben die Protokoll-Parameter, die in der Beschreibung von *flarp* auftauchen, Vorrang vor den Default-Werten. (Was es mit diesen Protokoll-Parametern auf sich hat, wird später verraten).

## Telefonnummer und Login-Information

In der Beschreibung eines Systems geben die Einträge `phone`, `call-login` und `call-password` die beim Einwählen in dieses System zu verwendende Telefonnummer, den Loginnamen und das Paßwort an. Wichtig ist außerdem noch das Gerät, das zum Wählen verwendet werden soll. Das wird mit dem `port`-Befehl angegeben. Da in diesem Beispiel nur ein Modem benutzt wird, kann diese Angabe in den Defaults-Teil geschrieben werden.

Sollten Sie mehrere Modems verwenden, können Sie den Port auch für jedes System einzeln angeben. Praktischer ist es dann allerdings, anstelle des Geräts mit dem `speed`-Befehl die gewünschte Übertragungsgeschwindigkeit anzugeben; UUCP übernimmt es dann, anhand dieser Information ein passendes Gerät auszuwählen. Das hat den Vorteil, daß UUCP unter mehreren vorhandenen Geräten mit passender Übertragungsrate ein freies auswählen kann.

## Erlaubte Zeiten

Mit dem Befehl `time` können Sie kontrollieren, wann ein System angerufen werden darf. Der Wert `Any` erlaubt Verbindungen zu jeder Zeit. Sie können allerdings auch kompliziertere Zeitangaben machen, wie beispielsweise `Mo-Fr0800-1700`; dies bezeichnet den Zeitraum von 8 Uhr bis 17 Uhr an Werktagen. Man kann dies auch durch `Wk0800-1700` abkürzen. Zeitangaben bestehen also aus einer Zeitspanne (in 24-Stunden-Schreibweise), und einer optionalen Angabe von Wochentagen (in den üblichen englischen Abkürzungen `Mo`, `Tu`, `We`, `Th`, `Fr`, `Sa` und `Su`).

Sie können auch mehrere Zeitangaben kombinieren, zum Beispiel `Wk2305-0755 , Sa , Su`; dies schließt die Nachtzeit an Wochentagen sowie das gesamte Wochenende mit ein.

Der zweite Parameter des `time`-Befehls ist optional und bezeichnet die Zeit, die *uucico* verstreichen läßt, bevor es einen erneuten Anruf bei dem Zielsystem zuläßt. Wenn *uucico* vorher aufgefordert wird, das

System anzurufen, wird es die Arbeit verweigern und in der Log-Datei die Meldung ``Retry time not reached" hinterlassen. Sie können *uucico* zwingen, das System trotzdem anzuwählen, indem Sie statt der Option `-s` die Option `-S` verwenden:

```
$ uucico -S cicero
$ _
```

UUCP merkt sich die Zeit des letzten Anrufs und den Status in privaten Dateien; Sie können diese Information mit dem folgenden Befehl anzeigen lassen:

```
$ uustat -m
cicero      12-06 19:25 Dial failed (3 tries, next after 12-06 19:28)
tauris      12-06 19:10 Conversation complete
$ _
```

## Das Chat-Skript

Sehr wichtig ist außerdem die `chat`-Information. Ein solches ``chat script" (zu deutsch: Tratsch-Anweisung) erklärt UUCP, wie es sich mit dem fremden System zu unterhalten hat, damit es sich einloggen darf. Es setzt sich aus mehreren Teilen zusammen, die abwechselnd beschreiben, auf welche Ausgabe des fremden Systems UUCP warten muß, und was es darauf zu antworten hat. Das oben angegebene Skript ist sehr simpel: es beschreibt im Grunde genau das, was auch Sie immer tun, wenn Sie sich bei Ihrem System anmelden: Am Login-Prompt geben Sie Ihre Benutzerkennung ein, warten auf den Paßwort-Prompt und geben Ihr Paßwort ein. In obigem Beispiel wartet UUCP auf die Aufforderung `ogin:` und schickt daraufhin die Benutzerkennung. Das Chat-Skript gibt diese Benutzerkennung nicht direkt an, sondern verwendet den Platzhalter `\L`, der bei der Ausführung von UUCP durch den Wert von der `call-login`-Angabe ersetzt wird. Anschließend wartet es auf die Aufforderung `ssword:`, und schickt daraufhin das Paßwort (symbolisiert durch `\P`).

Sie mögen sich jetzt vielleicht fragen, warum im Chat-Skript die Eingabeaufforderungen so verstümmelt auftauchen. Der Grund ist, daß das Skript unabhängig davon funktionieren soll, ob das fremde System `Login:` oder `login:` ausgibt. Beim Paßwort-Prompt hat man den zweiten Buchstaben aus ästhetischen Gründen dann gleich auch noch weggelassen.

Chat-Skripts können natürlich beliebig kompliziert werden, beispielsweise, wenn das andere System gelegentlich dazu gebracht werden muß, den Login-Prompt erneut auszugeben. Dies können Sie mit einer Art Verzweigung erreichen, die immer dann ausgeführt wird, wenn UUCP eine Weile vergeblich auf eine Äußerung des fremden Systems warten mußte. Eine Allround-Version des Chat-Skripts, das in den meisten Situationen funktionieren sollte, ist folgende:

```
chat      " " \r\n\c ogin:-BREAK-ogin: \L ssword: \P
```

Die erwähnte Verzweigung sehen Sie im Mittelteil: wartet UUCP vergeblich auf `ogin:`, so schickt es einen `BREAK` und wartet erneut auf `ogin:`. Erst wenn es diesen nach einer Weile immer noch nicht gesehen hat, gibt es dann auf.

Die ersten beiden Teile des Skripts sind für Systeme gedacht, die erst auf ein Zeichen des Anrufers warten, bevor sie ihren Prompt ausgeben; bei allen anderen Systemen sollte dies unschädlich sein. Die beiden Anführungszeichen bewirken, daß UUCP zunächst auf gar nichts wartet, sondern sofort das zweite Feld sendet: ein Carriage-Return (ASCII 13) und ein Linefeed (ASCII 10). Das Zeichen `\c` bewirkt, daß kein weiteres Linefeed ausgegeben wird, wie es normalerweise geschieht.

Zuletzt sei noch erwähnt, daß manche Systeme einen Moment Bedenkzeit benötigen, bevor man ihnen die

Benutzerkennung oder das Paßwort senden kann (z.B. SCO Unix). In solchem Falle sollten Sie eine kleine Verzögerung einbauen, indem Sie vor die Angaben \L und \P ein \d angeben.

## Protokoll-Parameter

Um sich an verschiedene Gegebenheiten anpassen zu können, bietet UUCP eine Auswahl von Protokollen zur Übertragung von Daten an. Traditionell werden diese Protokolle durch einen einzelnen Buchstaben benannt. Das ist zwar nicht sehr einprägsam, aber so viele sind's ja auch gar nicht.

Das wohl wichtigste dieser Protokolle ist das g-Protokoll. Es ist das älteste und am weitesten verbreitete UUCP-Protokoll, das eigentlich jede UUCP-Implementation akzentfrei sprechen sollte. Bei g handelt es sich um ein paketerorientiertes Protokoll, d.h. Daten werden nicht am Stück, sondern in einzelnen Paketen übertragen, die auf Fehlerfreiheit überprüft und im Bedarfsfalle erneut gesendet werden. Diese Eigenschaft hat zur Folge, daß g in hohem Maße für die Datenübertragung über Telefonleitungen geeignet ist.

Sie könnten nun an dieser Stelle einwenden, daß Ihr Modem aber in der Lage ist, Fehler zu erkennen und zu korrigieren, ist der Aufwand denn nötig? Ja, und zwar deshalb, weil das Protokoll des Modems zwar in der Lage ist, die meisten in der Telefonleitung auftretenden Fehler zu korrigieren, aber überhaupt keine Kontrolle darüber hat, was auf der Strecke zwischen Modem und Rechner passiert.

Für jedes fehlerfrei empfangene Paket sendet g eine Empfangsbestätigung (ACK) an den Absender; bleibt eine solche aus oder erhält der Sender stattdessen eine Fehlermeldung (NAK), geht er davon aus, daß das Paket bei der Übertragung verlorengegangen oder beschädigt wurde, und sendet es erneut. Wenn nun g für jedes einzelne Paket auf ein ACK warten müßte, wäre dies äußerst langsam, da ja beide Seiten die meiste Zeit nur Däumchen drehen würden. Deshalb kennt g ein "Fenster" (engl. *sliding window*), das es ihm erlaubt, eine Anzahl von Päckchen zu schicken, bevor es auf das Eintreffen des ersten ACK warten muß. Damit kann erreicht werden, daß die Leitung zu nahezu 100% ausgelastet wird, denn oft hat das empfangende System die Quittung für das erste Päckchen bereits geschickt, bevor der Sender das letzte im Fenster liegende Päckchen auf die Leitung geben konnte.

Ganz so rosig, wie die DFÜ-Welt zunächst aussieht, ist sie aber nicht: Wenn ein Päckchen im Fenster fehlerhaft ist, müssen alle Pakete des Fensters neu übertragen werden, egal, ob sie korrekt empfangen worden sind oder nicht. Die Wahrscheinlichkeit dafür, daß ein komplettes Fenster neu übertragen werden muß, hängt auch von der Größe des Fensters ab. Wenn immer die maximale Päckchen- und Fenster-Größe verwendet wird, kann es bei schlechten Verbindungen zu einem enormen Überhang doppelt übertragener Daten kommen.

Aus diesem Grund hatten früher die meisten UUCP-Varianten feste Werte hierfür vorgegeben, nämlich eine Paketgröße von 64 Bytes und ein Fenster von 3 Paketen. Die Taylor-Konfiguration erlaubt Ihnen hingegen, diese Werte an Ihre Gegebenheiten anzupassen. Hierzu dient das Schlüsselwort `protocol-parameter`. Das erste Argument ist immer der Protokoll-Name (z.B. g), gefolgt von dem zu setzenden Parameter. Welche Parameter überhaupt gesetzt werden können, und welche Werte zulässig sind, hängt natürlich von dem Protokoll ab. Dies ist sehr ausführlich in der offiziellen Dokumentation zu Taylor-UUCP beschrieben. Für unser Beispiel sind nur das g-Protokoll und dessen Paket- und Fenster-Größe interessant.

Die Paketgröße kann über den Parameter `packet-size` eingestellt werden. Erlaubt sind alle Zweierpotenzen zwischen 32 und 4096; die Voreinstellung liegt bei 64. Das Fenster wird über den Parameter `window` gesetzt und darf zwischen 1 und 7 liegen; Default ist 7.

Neben dem g-Protokoll versteht Taylor-UUCP unter anderem noch die Protokolle e und f, die zur Übertragung über TCP/IP-Verbindungen verwendet werden, G, das eine System V-spezifische Variante von g ist, und das neue i-Protokoll. Dieses Protokoll kann über eine Modemleitung gleichzeitig Pakete empfangen und senden: Protokolle wie g kennen zu jedem Zeitpunkt jeweils nur einen Sender und einen Empfänger (Master und Slave); ersterer schickt Datenpäckchen, und letzterer sendet nur die Bestätigungen (ACK und NAK) zurück, d.h. in der Richtung vom Slave zum Master ist die Leitung nur schlecht

ausgelastet. Diesen Mißstand behebt i, indem es die Unterscheidung zwischen Master und Slave aufhebt, und Datenpakete und Bestätigungen kombiniert. Ein weiterer Vorteil des i-Protokolls ist eine deutlich höhere Fenstergröße: sie kann bis auf 31 hochgesetzt werden.


## Die port-Datei


Damit UUCP Ihr Modem überhaupt ansprechen kann, müssen Sie die `port`-Datei entsprechend konfigurieren. Das ist schnell getan:

```
# /usr/lib/uucp/port
# Modem with V42bis compression
port          modem1
type          modem
device        /dev/cua1
speed         9600
dialer        hayes
```

Diese Datei enthält nur einen einzigen Eintrag, der das Modem am Port COM2 beschreibt. Ein solcher Eintrag wird durch das Schlüsselwort `port` eingeleitet, das dem Port einen eindeutigen Namen zuweist. Dies ist derselbe Name, wie Sie ihn in der Datei `sys` mit dem `port`-Befehl angegeben haben.

Die nächste Zeile benennt den Typ der Schnittstelle; anstelle von `modem` sind hier unter anderem Typen wie `direct` für eine Direktleitung, oder `tcp` für eine TCP/IP-basierte Verbindung zulässig.

Der Befehl `device` bezeichnet die Gerätedatei, durch die das Modem angesprochen werden soll. Unter UN\*X erfolgt der Zugriff auf Peripherie-Geräte über spezielle "Dateien" im Verzeichnis `/dev`. Linux verwendet für serielle Schnittstellen die Gerätedateien `cua0`, `cua1`, usw. bzw. `ttyS0`, `ttyS1`, etc. Dabei greifen die Gerätedateien mit der gleichen Nummer (minor number) auf dasselbe Gerät zu, nur auf unterschiedliche Art. `cua1` beispielsweise wird zum aktiven Ansprechen des Geräts verwendet, wie zur Anwahl eines anderen Rechners oder zum Konfigurieren des Modems; dagegen wird `ttyS1` im allgemeinen verwendet, um das Einloggen über die serielle Schnittstelle zu ermöglichen. 

Dabei entsprechen die Gerätedateien `cua0` bis `cua3` den Standard-Schnittstellen COM1 bis COM4.  Da Ihr Modem an COM2 angeschlossen ist, tragen Sie hier `/dev/cua1` ein.

Bitte beachten Sie, daß Sie hier auf jeden Fall den tatsächlichen Namen der Gerätedatei eintragen müssen. UUCP ignoriert symbolische Links, wie beispielsweise `/dev/modem`.

Der nächste Eintrag, `speed`, legt die Übertragungsgeschwindigkeit fest, die UUCP zur Kommunikation mit dem Modem benutzen soll. Da ein Modem mit 2400 Baud bei eingeschalteter V42bis-Kompression einen maximalen Datendurchsatz von 9600 Bit/s erreichen kann, tragen Sie hier 9600 ein. Damit sendet der Rechner zwar oft schneller, als das Modem die Daten auf die Leitung schicken kann, das ist aber nicht schlimm, wenn auf der seriellen Schnittstelle Hardware-Handshake eingeschaltet worden ist. So kann das Modem den Datenfluß vom Rechner regulieren. Schimmer wäre, wenn Sie diesen Wert zu niedrig wählen, weil dann der hohe Durchsatz auf der Modem-zu-Modem-Verbindung durch eine langsame Modem-zu-Rechner-Verbindung wieder zunichte gemacht wird.

Der letzte Eintrag teilt UUCP mit, um welchen Typ von Modem es sich bei dem angeschlossenen Gerät handelt. In diesem Fall ist dies ein Hayes-kompatibles Modem. Der Name, den Sie dabei vergeben, tut wenig zur Sache, da Sie die Beschreibung des Modems selber erstellen müssen.

# Die dial-Datei

Dies ist die letzte wichtige Datei. Sie beschreibt, wie UUCP das Modem veranlassen kann, die gewünschte Telefonnummer zu wählen. Die Konfiguration für das Hayes-Modem aus unserem Beispiel könnte beispielsweise folgendermaßen aussehen:

```
# /usr/lib/uucp/dial
# Dialer information for Hayes-compatible modem
dialer             hayes
chat               " " ATZ OK \dATV1E0Q0 OK \dATDP\D CONNECT
chat-fail          ERROR
chat-fail          BUSY
chat-fail          NO\sCARRIER
chat-fail          NO\sDIALTONE
dtr-toggle         true
```

Diese Datei enthält wiederum nur einen einzigen Eintrag, der den Modem-Typ `hayes` beschreibt. Im wesentlichen wird hier definiert, in welchen Bahnen die Unterhaltung zwischen UUCP und dem Modem verlaufen soll. Wieder wird dies in Form eines Chat-Skripts angegeben, wie Sie ihm bereits im Zusammenhang mit dem Einloggen im Abschnitt [7.4.8](#) begegnet sind.

Der wichtige Teil des Chat-Skripts wird durch den Befehl `chat` angegeben. In dem Beispiel wartet UUCP zunächst auf keinerlei Reaktion des Modems (d.h. den leeren String), sondern schickt von sich aus die Sequenz `ATZ`. Dies ist die Aufforderung an das Modem, sich zu initialisieren und interne Variablen mit vordefinierten Werten zu besetzen. Diese Sequenz sollte auf jeden Fall geschickt werden, damit sich das Modem in einem definierten Anfangszustand befindet.

Als Antwort auf dieses Kommando wartet UUCP auf den String `OK`. Erhält es diesen, sendet es den nächsten String, der mit einer kurzen Verzögerung (`\d`) beginnt, und anschließend den Befehl `ATV1E0Q0` absetzt. Dieser schaltet das lokale Modem-Echo aus, und stellt das Gerät auf Klartext-Antwortcodes ein. Wieder erwartet UUCP als Antwort hierauf `OK` und fordert das Modem anschließend auf, die gewünschte Telefonnummer zu wählen. Dies geschieht durch den Befehl `ATDP`, gefolgt von der eigentlichen Nummer. Anstelle von `\D` setzt UUCP die der `sys`-Datei entnommene Nummer ein. Zuletzt wartet es auf die Meldung `CONNECT`, mit der das Modem den erfolgreichen Verbindungsaufbau mit der Gegenstelle anzeigt.

Sollte UUCP innerhalb einer gewissen Zeit nicht den String finden, den das Chat-Skript angibt, so bricht es den Anwahlvorgang mit einer Fehlermeldung ab. Dieser Fehler wird in der Log-Datei als ```Timed out in modem chat``` vermerkt. Das kann beispielsweise passieren, wenn das Modem nicht eingeschaltet ist, aber auch, wenn beispielsweise die Gegenstelle besetzt ist. In diesem Fall liefert das Modem allerdings eine Fehlermeldung zurück, die im allgemeinen aufschlußreicher ist als ein bloßes ```Timed out```. Auf diese Fehlermeldungen können Sie UUCP trainieren, indem sie diese mit dem `chat-fail` Kommando angeben. Wann immer UUCP einen dieser `chat-fail` Strings erkennt, bricht es den Anwahlvorgang ab und vermerkt die erhaltene Fehlermeldung in der Log-Datei.

Die in unserem Beispiel angegebenen Fehlermeldungen werden von einem Hayes-kompatiblen Modem zurückgegeben, wenn etwa der Anschluß besetzt ist (`BUSY`) oder nach dem Abheben kein Freizeichen zu hören ist (`NO DIALTONE`). Das Zeichen `\s` in diesem String markiert ein Leerzeichen.

Der letzte Befehl in diesem Eintrag beginnt mit dem Schlüsselwort `dtr-toggle`. Dieser Befehl veranlaßt UUCP, die DTR-Leitung (Data Terminal Ready) der seriellen Schnittstelle auf `LOW` zu ziehen, bevor es das Modem anspricht. Dies ist für manche Modems notwendig, die an dem Zustand der DTR ablesen, ob die Rechnerseite sie ansprechen will.

# Testen der Konfiguration

Nachdem Sie UUCP nun installiert haben, sollten Sie das System testen, indem Sie eine Verbindung mit *cicero* aufzubauen versuchen. Sie führen dazu folgende Kommandos aus:

```
$ uucico -s cicero -x 2
$ tail -f /var/spool/uucp/.Admin/audit.local
uucico cicero - (1993-12-06 21:55:02.18 4172) Calling system cicero (port cua1)
uucico cicero - (1993-12-06 21:55:37.42 4172) Login successful
uucico cicero - (1993-12-06 21:55:38.02 4172) Handshake successful (protocol 'g
' sending packet/window 256/7 receiving 256/7)
uucico cicero - (1993-12-06 22:01:35.45 4172) Protocol 'g' packets: sent 3, re
sent 0, received 3
uucico cicero - (1993-12-06 22:01:40.58 4172) Call complete (1 seconds 0 bytes 0 bps)
```

Der zweite Befehl erlaubt Ihnen, den Aufbau der Verbindung zu verfolgen.

## Regelmäßige Verbindungen

Wenn Sie Ihr System einmal soweit konfiguriert haben, daß Mail und News einwandfrei funktionieren, können Sie dazu übergehen, UUCP unbeaufsichtigt laufen zu lassen. Dazu können Sie beispielsweise für den Benutzer *uucp* einen *cron*-Auftrag anlegen, der zu festgelegten Zeiten *uucico* aufruft oder alte Log-Dateien wegwirft. Zum Beispiel könnten Sie folgendes in eine Datei *crontab.uucp* eintragen:

```
PATH=/usr/lib/uucp
0 22 * * * uucico -s cicero
0 6 * * * uucico -s cicero
```

Dies würde jeweils um 6 Uhr und 22 Uhr *uucico* aufrufen, um Daten mit *cicero* auszutauschen. Sie können den *cron*-Auftrag anschließend als Superuser mit folgendem Befehl installieren:

```
# crontab -u uucp -r crontab.uucp
```

Diese Lösung ist allerdings noch nicht ganz befriedigend, denn es ist möglich, daß bei *cicero* beispielsweise kurzzeitig besetzt ist; Ihr Anruf um 22 Uhr würde beispielsweise fehlschlagen, obwohl er vielleicht eine Minute später Erfolg gehabt hätte. Die Lösung ist hier, *uucico* mehrfach im Abstand von wenigen Minuten aufzurufen. Damit dies korrekt funktioniert, müssen Sie aber in der *sys*-Datei bei *cicero* noch eine Änderung vornehmen. Entweder für das in Frage stehende System oder im Defaults-Teil der *sys*-Datei tragen Sie die Option

```
success-wait          900
```

ein. Dies verhindert, daß *uucico* nach einer erfolgreichen Verbindung sofort wieder anruft. Das obige Beispiel setzt die Wartezeit auf 15 Minuten (gleich 900 Sekunden). Jetzt können Sie den *cron*-Job abändern, so daß *uucico* mehrere Male in Folge aufgerufen wird, aber höchstens einmal die Verbindung aufbaut:

```
PATH=/usr/lib/uucp
0,2,4,6 22 * * * uucico -s cicero
0,2,4,6 6 * * * uucico -s cicero
```

---

[Next](#) [Up](#) [Previous](#) [Contents](#) [Index](#)

**Next:** [Elektronische Post mit smail](#) **Up:** [Datenreisen und reisende Daten](#) **Previous:** [Seriell einloggen](#)





## Subsections

- [Wie sieht eine Mail denn nun aus?](#)
  - [Adressen, Adressen, Adressen](#)
  - [Taler, Taler, Du mußt wandern](#)
  - [Email-Software unter Linux](#)
  - [Installation von \*smail\*](#)
    - [Die config-Datei](#)
  - [Elektronische Post mit \*elm\*](#)
    - [elm-Konfiguration](#)
  - [Ein Test](#)
- 

# Elektronische Post mit *smail*

Einer der am weitesten verbreiteten Dienste, die durch die Vernetzung von Rechnern möglich werden, ist die elektronische Post, meist *Email* genannt. Email erlaubt es Benutzern auf unterschiedlichen Rechnern - auch auf entgegengesetzten Seiten des Erdballs - miteinander zu kommunizieren, und zwar erheblich schneller, als es mit gewöhnlicher Post jemals möglich wäre. Im Internet benötigt eine Botschaft oft nur wenige Minuten von Europa in die USA. In reinen UUCP-Netzen ist ein elektronischer Brief natürlich wesentlich länger unterwegs, da er die meiste Zeit seiner Reise auf den Spool-Platten diverser Rechner verbringt. Trotzdem kann sich die Beförderungsgeschwindigkeit immer noch mit der der Deutschen Bundespost messen.

Zur Beförderung von Email sind natürlich Standards vonnöten, damit Maschinen, die Botschaften miteinander austauschen, einander überhaupt verstehen. Ein solcher Standard ist RFC 822, der das Format von Mails im Internet regelt. Da viele UUCP-Netze eng an das Internet angebunden sind, hat sich RFC 822 auch in diesen weitgehend durchgesetzt. Wie es mit Standards so geht, hat natürlich jedes zweite Netz seinen eigenen. Wir werden uns hier jedoch nur mit RFC 822 beschäftigen.

## Wie sieht eine Mail denn nun aus?

Eine Mail ist im wesentlichen eine Datei, die den Text Ihres Briefes enthält. Ein Teil dieser Datei besteht aus administrativen Daten, wie den Adressen von Absender und Empfängern, die, ganz wie im richtigen Leben, im Kopf des Briefes untergebracht sind. Wir werden diesen Teil im folgenden allgemein mit *Kopf* bezeichnen, oder mit dem englischen Ausdruck *message header*; den eigentlichen Text der Botschaft nennen wir *Rumpf* oder *message body*.

Ein elektronischer Brief könnte beispielsweise so aussehen:

```
From fluxus.in-berlin.de!susanne Thu Jul 15 09:16:21 1994 remote from fluxus
Return-Path: <fluxus.in-berlin.de!cicero.in-berlin.de!susanne>
Received: from cicero.in-berlin.de by isis.in-berlin.de with uucp
        (Smail3.1.28.1 #6) id m0oGNY2-0000H9B; Thu, 15 Jul 94 10:16 MET DST
Received: by cicero.in-berlin.de from fluxus.in-berlin.de
        msg-id m0zF9AR.000G2Za; Thu, 15 Jul 94 09:00 MET DST
Received: by fluxus.in-berlin.de
        id AA0043n; Tue, 15 Jul 94 08:53:33 CET
Message-Id: <9407130840.AA02871@fluxus.in-berlin.de>
```

Date: Tue, 15 Jul 94 08:53:32 MESZ  
From: Susanne Bois <susanne@fluxus.in-berlin.de>  
To: karla@isis.in-berlin.de (Karla Kosolowski)  
Subject: Linux 3.1 draussen

Hi, Karla,

seit gestern läuft bei mir auch Linux 3.1. Ich kann Dir  
mal ein Band bespielen und es rueberschieben, wenn Du willst.

Susanne

Der Kopf des Briefes - der gesamte Bereich bis zur ersten Leerzeile - enthält die administrativen Informationen, die in einzelne Felder aufgeteilt sind. Jedes Feld beginnt mit einem Namen, gefolgt von einem Doppelpunkt, und dem eigentlichen Inhalt des Feldes. Dieses Feld kann auch auf der nächsten Zeile fortgesetzt werden, wenn die Zeile mit einem Einschub (TAB) beginnt. Einige der Informationen sind eher technischer Art, wie die `Received:` Felder, andere sind durchaus auch für die Empfängerin von Interesse, wie die Zeile `Date:`, die das Datum der Erstellung enthält, und `Subject:`, in der die Absenderin dem Brief eine Art Betreff-Zeile voranstellt. Die meisten dieser Felder werden von der Mail-Software automagisch ausgefüllt, beispielsweise Datum und Absenderadresse.

## Adressen, Adressen, Adressen


Wenn Sie einem Menschen einen Brief oder eine Postkarte schreiben, werden Sie diese natürlich mit dessen Anschrift versehen, bestehend aus Namen, Straße und Hausnummer sowie Wohnort. Auf ähnliche Weise müssen Sie natürlich auch der Transport-Software mitteilen, wie der Empfänger Ihres elektronischen Briefs zu erreichen ist. Verschiedene Netze haben dafür natürlich auch unterschiedliche Adressierungsarten und -formate. Uns werden aber nur zwei Formate interessieren: der traditionelle UUCP *bang path*, oder !-Pfad, und das RFC 822-Format.

Beiden Adreßformaten ist gemeinsam, daß sie aus je einer Benutzer- und einer Rechnerbezeichnung bestehen; d. h. ein Teil der Adresse beschreibt den Empfänger, meist durch dessen Benutzerkennung, während der andere den Zielrechner beschreibt, auf dem die Person überhaupt zu erreichen ist. Im obigen Beispiel bestand `karla@isis.in-berlin.de` aus den Teilen *isis.in-berlin.de*, dem vollständigen Namen des Rechners, und `karla`, dem Login-Namen von Karla. Beide werden durch das ``@''-Symbol voneinander getrennt. Diese Adressen-Schreibweise entspricht den Vorschriften von RFC 822.

Dem !-Pfad sind wir bereits im vorigen Abschnitt bei der Einführung in die Welt von UUCP begegnet, wo *fluxus!tauris* eine Abfolge von Rechnern bezeichnete, über die ein bestimmter Auftrag übermittelt werden sollte. Diese Schreibweise wurde früher in UUCP-Netzen auch häufig zur Adressierung von Mail benutzt und findet heute noch innerhalb der Transport-Software Verwendung. Dabei wird, getreu dem Motto ``Der Weg ist das Ziel'', das Zielsystem nicht durch einen eindeutigen Namen angegeben, sondern durch eine Auflistung der Systeme, über die es erreichbar ist. Um beispielsweise einen Brief an Jens auf dessen Maschine *tauris* zu senden, müßte Karla die Adresse `fluxus!tauris!jens` verwenden. Eine solche Adresse teilt der Email-Software mit, daß sie die Nachricht an *fluxus* übermitteln soll, das diese an *tauris* weiterreichen wird, wo sie an den Benutzer *jens* ausgeliefert werden wird. Diese Form der Adressierung wird jedoch nur noch äußerst selten verwendet, weswegen wir uns an dieser Stelle nicht weiter mit ihr beschäftigen werden.

Sie sehen in diesen Beispielen auch zwei Arten, ein System zu benennen: einerseits mit seinem einfachen ``Vornamen'', wie *isis*, und mit seinem vollen Namen, *isis.in-berlin.de*. Letzterer wird auch als der *kanonische Hostname* bezeichnet. Den zusätzlichen Teil des Namens (*in-berlin.de*) nennt man die *Domain*, d. h. den Bereich, dem der Rechner zugehörig ist. Domains wurden eingeführt, als dank der stark ansteigenden Zahl von vernetzten Rechnern sinnvolle Namen knapp zu werden begannen, und es auch immer schwieriger wurde, den Überblick über die Netzwerk-Topologie zu behalten. Die Idee von Domains ist, größere Gruppen von Systemen, die geographisch oder organisatorisch eng verbunden sind, in ``Meta-Systeme'' zusammenzufassen und mit einem gemeinsamen Gruppennamen zu belegen. Dies erleichtert unter anderem das Mail-Routing ganz ungemein, spielt aber auch in anderen Bereichen eine Rolle.

# Taler, Taler, Du mußt wandern

Um in einem Netzwerk eine Botschaft von einem System zu einem anderen schicken zu können, muß die Transport-Software wissen, auf welchem Weg sie diese befördern kann. Die Aufgabe, einen korrekten und möglichst optimalen Pfad zu finden, wird als *Routing* bezeichnet. In Lokalen Netzen (LANs) und dem Internet ist dies im allgemeinen recht einfach, da der Zielrechner meist direkt angesprochen werden kann. 

In UUCP-Netzen ist das schon kniffliger, da die Botschaft dabei durch die Hände mehrerer Systeme gereicht werden muß. Die einfachste Lösung ist der !-Pfad, der dem Benutzer die knifflige Aufgabe überläßt, einen zuverlässigen Pfad vom lokalen System zum Zielsystem zu finden. Diese Lösung ist natürlich unbefriedigend; es wäre angenehmer, wenn das System diese Pfade selbständig generieren würde, wenn man ihm nur den Zielrechner angibt.

Dies kann beispielsweise durch die Verwendung einer sogenannten *pathalias*-Datei erreicht werden. In einer solchen Datei werden Rechnern oder ganzen Domains !-Pfade zugeordnet, über die sie erreichbar sind. Wenn Sie Ihren gesamten Verkehr über nur ein UUCP-System abwickeln, werden Sie eine solche Datei allerdings nie benötigen, da es noch eine dritte Möglichkeit gibt - nämlich die Arbeit auf andere abzuwälzen.

Bei dieser Methode stellt sich Ihre Email-Software (absichtlich) dumm, und überläßt es dem intelligenteren System, aus den Adressen der Zielrechner vernünftige Pfade zu erzeugen. Der gängige Begriff hierfür ist *smart-host routing*.

## Email-Software unter Linux

In der UN\*X-Welt gibt es verschiedene Programme zur Erstellung und Beförderung von Email. Eins der bekanntesten Benutzer-Programme ist *elm*, das Ihnen ein Cursor-gesteuertes Menü zur Verfügung stellt, von dem aus Sie Botschaften versenden und die Nachrichten in Ihrem Briefkasten lesen können. *elm* wird in einem der folgenden Abschnitte näher beschrieben. Daneben gibt es noch viele weitere Programme, die alle ungefähr dieselbe Funktionalität bieten, jedoch in unterschiedlichen Graden der Benutzerfreundlichkeit.

Natürlich ist die Benutzer-Software nur der eine Teil des Post-Systems; der andere Teil ist die Transport-Software. Auch hier gibt es verschiedene Pakete, beispielsweise *sendmail*. Das wurde ursprünglich für das Unix der Universität in Berkeley geschrieben, und auf den meisten UN\*X-Plattformen verwendet. Für Linux existiert eine Portierung des *sendmail-5.56c* mit den IDA-Erweiterungen. Der neue *sendmail-8.6.9* schließlich läßt sich ganz ohne Probleme auf Linux compilieren.

*sendmail* steht allerdings in dem (früher sicherlich berechtigten) Ruf, für Anfänger schwer installierbar zu sein, weshalb viele ein anderes Paket für ihren Email-Transport verwenden: *smail-3.1.28*. Dieses Programm wurde von Landon Curt Noll und Robert S. Karr geschrieben und stellt ausreichende Funktionalität für kleinere bis mittlere Systeme zur Verfügung. Im folgenden Abschnitt werden die notwendigen Schritte für die Installation von *smail* auf einem UUCP-System dargelegt.

## Installation von *smail*

Abhängig von der Linux-Distribution, die Sie verwenden, finden Sie *smail* samt einiger zugehöriger Programme entweder in `/usr/bin` oder `/usr/local/bin`. Außerdem muß eine Kopie von *smail* als `/usr/lib/sendmail` vorhanden sein. Ferner benutzt *smail* eine oder mehrere Konfigurationsdateien, die allesamt in `/usr/lib/smail` beziehungsweise `/usr/local/lib/smail` abgelegt werden. Falls Sie sich nicht sicher sind, in welchem Verzeichnis diese Dateien abgelegt werden, können Sie das Verzeichnis mit folgendem Befehl erfragen:

```
smail -bP smail_lib_dir
```

Wir werden im folgenden die Konfiguration für ein System besprechen, das mit nur einem weiteren System per UUCP kommuniziert. Natürlich bleibt unser Beispiel weit hinter den Möglichkeiten von *smail* zurück; mehr darüber können Sie in den Manual-Seiten zu *smail* erfahren, sowie im Linux Networking Guide.

# Die config-Datei

Die wichtigste Konfigurations-Datei ist `config`. Für Karlas System könnte die Datei beispielsweise folgendermaßen aussehen:

```
#
# config file for isis.in-berlin.de
#
hostnames=isis.in-berlin.de
visible_name=isis.in-berlin.de
uucp_name=isis.in-berlin.de
#
smart_path=fluxus
smart_transport=uux
#
error_copy_postmaster
```

Zeilen, die mit einem Doppelkreuz beginnen, sind Kommentare und werden ignoriert. Alle Einträge in der Datei besetzen verschiedene Konfigurations-Variablen. Die ersten drei Variablen legen den Namen des Systems für verschiedene Operationen fest; in diesem Fall ist dies immer *isis.in-berlin.de*. Die genauen Bedeutungen dieser Variablen können Sie der Dokumentation entnehmen. Im Falle eines einzelnen Computers genügt es, wenn Sie alle drei Variablen mit dem kanonischen Namen Ihres Systems besetzen.

Die Variablen `smart_path` und `smart_transport` beziehen sich auf das oben vorgestellte smart-host routing. Erstere enthält den UUCP-Namen des Rechners, über den Sie alle ausgehenden Nachrichten ausliefern. In Karlas Fall geht alle Post über *fluxus*. Die zweite Variable teilt *smail* den Transport mit, über den dies erfolgen soll; ein Wert von *uux* bezeichnet die Auslieferung über UUCP an das Programm *rmail* auf dem nächsten Rechner, also *fluxus*. Dies ist die Standard-Methode, um in einem UUCP-Netz Post zu transportieren.

Die letzte Variable, `error_copy_postmaster`, ist eine Boolesche Variable. Wird sie wie in der Beispieldatei gesetzt, ☒ wird der Benutzer `postmaster` von jedem Fehler bei der Auslieferung einer Botschaft benachrichtigt und erhält eine Kopie derselben.

Neben `config` kennt *smail* noch einige weitere Konfigurationsdateien, die aber zum Funktionieren einer einfachen Installation nicht nötig sind: `routers`, `directors` und `transports`, die Details des Routing, der lokalen Auslieferung sowie des Transports zu anderen Systemen regeln. Wenn diese Dateien nicht existieren, verwendet *smail* sinnvolle Default-Werte.

## Elektronische Post mit *elm*

In der UN\*X-Welt gibt es verschiedene Programme zur Erstellung und Beförderung von Email. Eins der bekanntesten Benutzer-Programme ist *elm*, das Ihnen ein Cursor-gesteuertes Menü zur Verfügung stellt, von dem aus Sie Botschaften versenden und die Nachrichten in Ihrem Briefkasten lesen können. Die Oberfläche von *elm* könnte sich Ihnen beispielweise darstellen wie in Abb. [7.11](#) gezeigt.

Sie sehen hier eine mäßig gefüllte Mailbox; in der Mitte werden die Nachrichten der aktuellen Mailbox mit laufender Nummer, Absender, Zeilenzahl und Betreff-Zeile angezeigt. Der Pfeil am linken Rand markiert die aktuelle Nachricht, auf die sich alle der in dem kleinen Hilfsmenü angezeigten Funktionen beziehen. Wenn Sie im obigen Beispiel an der Eingabeaufforderung `RETURN` eingeben, wird *elm* die Nachricht eines gewissen Sebastian Hetze anzeigen. Wenn Sie *elm* das erste Mal starten, wird Ihre Mailbox allerdings im Allgemeinen leer sein.

```
Mailbox is '/usr/mail/okir' with 22 messages [ELM 2.4 PL17]
```

```
12 Jul 28 Vince Skahan      (128) Re: Some questions for the NAG
13 Jul 28 Vince Skahan      (1256) Linux news/mail/uucp compilation
```

```

14 Aug 4 Juergen Unger (273) Re: UUCP config (anon uucp)
15 Aug 5 Vince Skahan (58) Re: anon UUCP.
16 Oct 9 Barry Flanagan (85) Re: your mail
17 Oct 21 E. Marinyak (79) Network configuring
18 Oct 25 E. Marinyak (79) Re: Network configuring
19 Nov 9 Shyh-Horng Jou (58)
20 Nov 24 Wolfgang Michaelis (54) Re: Neues Release (war nix)
21 Nov 28 obarriga@abello.se (42) Okie dokie
-> 22 Dec 2 Sebastian Hetze (38) Telefonnummer

```

You can use any of the following commands by pressing the first character;  
d)delete or u)ndelete mail, m)ail a message, r)eply or f)orward mail, q)uit  
To read a message, press <return>. j = move down, k = move up, ? = help

Command: \_

### Hauptmenü des Mail-Readers *elm*.

Um mit dem Programm vertraut zu werden, können Sie zunächst ein wenig herumexperimentieren. Haben Sie bereits *smail* installiert, so können Sie jetzt nämlich bereits Post an andere Benutzer auf Ihrem Rechner verschicken, ohne daß weiterer Aufwand dazu nötig wäre.

Sie starten dazu *elm* von der Shell aus. Wenn Sie *elm* zum allerersten Mal aufrufen, werden Sie danach gefragt, ob *elm* zwei Verzeichnisse in Ihrem Benutzer-Verzeichnis anlegen darf; antworten Sie hierauf mit ``y(es)". Dann erscheint das Hauptmenü und zeigt aller Wahrscheinlichkeit nach eine leere Mailbox.

Wollen Sie nun eine Botschaft schreiben und abschicken, geben Sie an der Eingabeaufforderung *w* ein, und werden nacheinander nach der Adresse des Empfängers, der Betreff-Zeile, und eventuellen weiteren Empfängern gefragt. Im folgenden Beispiel schicken Sie eine Test-Botschaft an sich selbst; Ihre Eingaben sind kursiv gesetzt:

```

Command: Mail
Send the message to: karla
Subject of message: Test
Copies to: <Return>
Invoking editor...

```

Anschließend ruft *elm* einen Editor auf, in dem Sie Ihren Brief erstellen. Der voreingestellte Editor ist meist *vi*. Da dies nicht jedermanns Sache ist, stellen wir weiter unten eine Möglichkeit vor, dies zu ändern.

Nachdem Sie Ihren Brief geschrieben, abgespeichert und den Editor verlassen haben, stellt *elm* Ihnen folgende Frage:

```

Please choose one of the following options by parenthesized letter: s
e)dit message, edit h)eaders, s)end it, or f)orget it.

```

Um den Brief tatsächlich abzuschicken, drücken Sie entweder *s* oder *RETURN*. Die Option *e* bringt Sie wieder in den Editor zurück, und *f* wirft den Brief weg (falls Sie sich anders entschieden haben sollten). Mit der Option *h* gelangen Sie in ein Menü, in dem Sie den Kopf der Botschaft editieren können; dort können Sie beispielsweise die Liste der Empfänger oder die Betreff-Zeile verändern.

Nachdem Sie Ihren Brief abgeschickt haben, sollte Ihre Festplatte etwas rattern, und zunächst wieder die unverändert leere Mailbox dargestellt werden. Das liegt daran, daß *smail* Ihren Brief im Hintergrund ausliefert. Nachdem das Rattern der Platte dann etwas nachgelassen hat, sollten Sie irgendeine Taste drücken (beispielsweise eine Cursor-Taste); *elm* nimmt dies zum Anlaß, nachzuprüfen, ob neue Post eingetroffen ist. Ist alles gutgegangen, erscheint in Ihrer Mailbox jetzt Ihre Testnachricht, und Sie können sie anzeigen lassen, indem Sie die Return-Taste drücken.

# elm-Konfiguration

Um mit *elm* auch Post über das Netz verschicken zu können, müssen Sie ihn ebenfalls noch konfigurieren. Die *elm*-eigene Konfigurations-Datei befindet sich meistens im Verzeichnis `/usr/lib/elm` oder `/usr/lib`, und heißt `elm.rc`. Für unser Beispiel-System *isis* könnte sie beispielsweise so aussehen:

```
# elm.rc
#
hostname      = isis
hostdomain    = .in-berlin.de
hostfullname  = isis.in-berlin.de
#
# Transport von Umlauten etc.
charset       = iso-8859-1
displaycharset= iso-8859-1
textencoding  = 8bit
#
# Editor für Mails
editor        = /usr/bin/emacs
```

Die ersten drei Befehle teilen *elm* den Namen des lokalen Systems mit, aufgespalten in den eigentlichen Rechner-Namen und den Domain-Namen.

Die folgenden drei Variablen erzeugen Informationen im Mail-Kopf, die helfen sollen, Sonderzeichen heil durch das Netz zu manövrieren. Im US-ASCII-Standard-Zeichensatz sind nämlich Zeichen wie die uns so vertrauten Umlaute keineswegs vorhanden; es gibt sie nur in erweiterten Zeichensätzen, wie beispielsweise dem von DOS bekannten IBM-Zeichensatz. Linux benutzt zur Darstellung von internationalen Zeichen einen anderen Standard, bekannt unter dem Namen ISO-8859-1. Um der transportierenden und empfangenden Mail-Software mitzuteilen, daß eventuell in der Nachricht auftauchende Umlaute in diesem Standard kodiert sind, kann *elm* diese Information in den Mail-Kopf einfügen.

Die letzte Variable ist die versprochene Methode, den *vi*-Editor durch einen anderen Editor zu ersetzen, hier durch EMACS.

Parameter, die Sie in der Datei `elm.rc` einstellen, gelten für alle Benutzer. Nun kann es aber vorkommen, daß manche Benutzer *elm* gerne anders konfigurieren würden. Das ist grundsätzlich möglich, da *elm* neben der globalen Konfigurations-Datei auch noch eine private Konfigurations-Datei kennt, nämlich `.elm/elmrc` in Ihrem Heimatverzeichnis. Sie können diese Werte in dieser Datei auch verändern, indem Sie in *elm* das Konfigurations-Menü aufrufen (mit dem Befehl `o`) und die veränderten Werte anschließend abspeichern.

## Ein Test

Um zu testen, ob Ihre Konfiguration korrekt ist, versuchen Sie, eine Nachricht an einen Benutzer auf einem anderen System zu schicken, beispielsweise an die Administratoren des UUCP-Systems, an das Sie sich angeschlossen haben. Wenn Sie die Nachricht mit *elm* erstellen, geben Sie nach der Eingabeaufforderung ```Send the message to:``` die volle Adresse an, z.B. `root@cicero.in-berlin.de`.

Nachdem Sie die Nachricht geschrieben und abgeschickt haben, können Sie mit folgendem Befehl überprüfen, ob sie auch tatsächlich an UUCP weitergegeben worden ist:

```
$ uustat -a
ciceroC0001 cicero " " 12-06 22:39 Executing rmail root (sending 436 bytes)
```

Die Ausgabe sollte ähnlich wie die zweite Zeile dieses Beispiels aussehen. Sie zeigt an, daß für das System *cicero*

der Auftrag aufgegeben wurde, den Befehl ``*rmail root*" auszuführen, und eine zusätzliche Datei von 436 Bytes - die eigentliche Mail - zu übertragen.

Sollte der Befehl *uustat* allerdings keine Ausgabe zeigen, ist etwas schiefgelaufen. In diesem Falle finden Sie im Verzeichnis `/var/spool/smtp/log` die Log-Dateien `logfile` und `paniclog`; diese sollten aufschlußreiche Fehlermeldungen enthalten. Falls der Fehler innerhalb der UUCP-Konfiguration zu suchen ist, können Sie auch die entsprechenden UUCP-Log-Files zu Rate ziehen. Sollten Sie mit Hilfe dieser Fehlermeldungen nicht in der Lage sein, den Fehler zu beheben, hilft wohl nur noch, zu weiterführender Dokumentation zu greifen.

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [Usenet News](#) **Up:** [Datenreisen und reisende Daten](#) **Previous:** [UUCP - Das Internet der](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

Next: [INN](#) Up: [Datenreisen und reisende Daten](#) Previous: [Elektronische Post mit smail](#)

## Subsections

- [Die technischen Details](#)
  - [News-Software](#)
- 

# Usenet News

Wenn Kommunikation im Internet sich nur auf elektronische Mail beschränkte, wäre die ganze Sache auf die Dauer recht langweilig. Die richtige Würze kommt erst durch einen weiteren Netzdienst hinzu, die Usenet News. Sie stellen so etwas wie ein weltweites *Forum Romanum* dar, wo jede Netzbürgerin und jeder Netzbürger zensuriert ihre Meinung abgeben, Fragen stellen oder Information verbreiten können.


Diskussionen finden in sogenannten Newsgruppen statt, die nach Themen organisiert sind und ähnlich wie eine Wandzeitung funktionieren. Teilnehmer, die einen Beitrag leisten wollen, schreiben einen meist kurzen Text (Artikel genannt) und speisen ihn ins Usenet ein, das ihn dann weltweit verteilt.

Die Themen der einzelnen Gruppen reichen vom Technischen über Politik bis zum puren Nonsens. Umgangston und Informationsgehalt unterscheiden sich zwischen einzelnen Gruppen teilweise dramatisch. Wenn in manchen Gruppen täglich an die hundert Artikel erscheinen, verwundert es kaum, daß aus manchen Diskussionen ein wüstes Durcheinander wird. Und es gibt Gruppen mit wesentlich höherer Beteiligung.

Die genaue Größe des Usenet kennt niemand, sie läßt sich nur schätzen. Es gibt derzeit an die 5000 Newsgruppen, in denen monatlich um die 3 Millionen Artikel erscheinen. Das Gesamtvolumen bewegt sich auf die 200 Megabytes *täglich* zu...

Um ansatzweise den Überblick behalten zu können, sind die Namen der Newsgruppen hierarchisch aufgebaut, so daß man schnell erkennen kann, um was es sich dabei dreht. Betrachten Sie die Gruppe `comp.os.linux.announce`: Sie ist Teil der Hierarchie `comp`, in der fast alle computer-orientierten Gruppen angesiedelt sind. Die Unterhierarchie `comp.os` ist für Newsgruppen da, die sich mit Betriebssystemen befassen (`os` ist die gängige Abkürzung für Operating System). Unsere Newsgruppe ist also eine von mehreren Gruppen für Linux-Benutzer und enthält wichtige Ankündigungen und ähnliches.


Neben den sieben großen Hierarchien `comp`, `misc`, `news`, `rec`, `sci`, `soc` und `talk`, in denen Englisch die Verkehrssprache ist, gibt es noch eine Reihe "nationaler" Hierarchien, z.B. `de` für den deutschsprachigen Raum, `fr` für Frankreich und `fr.j` für Japan.

Das Einrichten (und ganz selten auch Löschen) von Newsgruppen folgt demokratischen Spielregeln: nach einer vorgeschriebenen Diskussionsphase wird abgestimmt; stimmberechtigt ist jede Person, die am Usenet teilnimmt und in der Lage ist, eine elektronische Mail abzuschicken. Die genauen Modalitäten können sich allerdings von Hierarchie zu Hierarchie unterscheiden; im deutschen Usenet gelten beispielsweise etwas andere Regeln als in den oben erwähnten "Big Seven." Daneben gibt es auch noch die `alt`-Hierarchie,  in der das Anlegen von Gruppen fast völlig unreglementiert ist; hier finden sich Nischengruppen, in der eine Handvoll Enthusiasten debattiert (die erste Linux-Gruppe hieß `alt.os.linux`), kontroverse Gruppen wie `alt.sex.*`, und nicht zuletzt Bandbreitenvernichter wie `alt.binaries.pictures.*`, in denen digitalisierte Bilder von mehr oder minder zweifelhaftem



Wert ausgetauscht werden.

Das Usenet ist sicherlich das größte und freieste Kommunikationsmedium unseres Planeten. Solange Sie einen Zugang zum Netz haben, können Sie hier Ihre Meinung unzensuriert verbreiten. Die Grenzen der Toleranz werden allein durch die anderen Netzteilnehmer gesetzt, die Beschimpfungen und Beleidigungen entsprechend erwidern. Jeder Versuch einer äußeren Einflußnahme oder Zensur wird energisch bekämpft.

Das steht nicht im Widerspruch zur Existenz sogenannter moderierter Gruppen. In solchen Gruppen können Sie nicht ohne weiteres einen Artikel posten,  sondern müssen ihn vorher an die Moderatorin oder den Moderator schicken, die den Artikel entweder postet oder Ihnen zurückschickt. Diese Einschränkung der Redefreiheit wird wiederum im Usenet demokratisch beschlossen; sie dient dazu, den Anteil an unsinnigen, unpassenden oder allzu provokativen Artikeln herauszufiltern. Oft werden Gruppen mit sehr kontrovers diskutierten Themen moderiert, wie `soc.feminism`. Daneben gibt es auch eine ganze Reihe von moderierten technischen Gruppen, z.B. `comp.os.linux.announce`.

## Die technischen Details

Usenet ist der Oberbegriff für den Verbund aller Rechner, die News miteinander austauschen. Die Definition ist, zugegeben, etwas redundant; aber es ist bisher noch niemandem etwas besseres eingefallen (und es interessiert wohl auch niemanden, solange es funktioniert). Das Usenet ist keine Institution oder Netz im eigentlichen Sinne - es gibt keine zentrale Autorität, keine eigene Infrastruktur und kein favorisiertes Betriebssystem. Einziges Merkmal einer Usenet-Site ist eben, daß sie News transportiert.

Die einzelnen Nachrichten im Usenet werden Artikel genannt und sind sehr ähnlich wie eine Mail aufgebaut. Ein Artikel besteht aus einem Kopf (*Header*), der wichtige Informationen für die Verarbeitung des Artikels enthält, und dem eigentlichen Text (dem *Body*). Ein typischer Artikel sieht so aus:

```
Path: flarp!zib-berlin.de!news.mathworks.com!uunet!nsu.edu!not-for-mail
From: joe.doe@cs.nsu.edu (Joe Doe)
Newsgroups: news.software.b
Subject: Posting problem
Date: 29 Feb 1995 10:12:21 -0500
Organisation: Nebraska State University
Message-ID: <3hj$km6lej@news.nsu.edu>
```

Hi,

I installed INN 1.4 just yesterday, and everything works fine except for postings by local users. Whenever I post an article from tin, trn, or nn, it vanishes into the great bit-bucket. Anybody got any idea?

Advance thanks  
Joe

Man sieht hier neben den vom Mail-System bekannten Feldern wie `From:` und `Date:` auch einige neue wie beispielsweise `Newsgroups:`. Diese Zeile bezeichnet die Newsgroups, in die der Artikel gepostet wurde. Um die anderen Felder zu erklären, muß ich etwas weiter ausholen.

News werden dezentral verteilt. Wenn Sie einen Artikel schreiben und ins Netz einspeisen (im Jargon *posten* genannt), wird der Artikel von Ihrem Newssystem an alle interessierten Nachbar-Sites

weitergegeben; die wiederum reichen ihn an ihre Nachbarn weiter, usf. Es ist einleuchtend, daß diese Methode für eine schnelle Verbreitung der Information sorgt, und gleichzeitig sehr unempfindlich gegenüber Störungen im Netz ist. ☒ Andererseits muß ein System in der Lage sein, Duplikate zu vermeiden, bzw. sie auszusortieren, falls sie doch einmal auftreten. Dazu dienen die Header-Zeilen `Message-ID:` und `Path:`.

Das erste dieser beiden Felder enthält eine weltweit eindeutige Kennung des Artikels; indem sich ein System nun die IDs aller Artikel merkt, die es innerhalb der letzten n Tage bearbeitet hat, kann es Duplikate sofort aussortieren. Das `Path:`-Feld erfüllt einen ähnlichen Zweck; es enthält die Liste aller Systeme, die der Artikel auf dem Weg vom Absendersystem bis zu uns bereits durchlaufen hat. Damit kann das Newssystem bereits vorab einige Systeme aussortieren, an die es den Artikel nicht mehr weitergeben muß - zumindest nämlich dasjenige System, von dem es den Artikel gerade erhalten hat.

Bevor wir uns der Software-Seite des Usenet zuwenden, möchte ich noch ein paar Worte über den Transport der News verlieren. Wir haben bereits gesehen, daß Artikel von jedem System an dessen Nachbarn weitergereicht werden. Solch ein ``Datenkanal'' zwischen zwei Systemen nennt sich ein *Feed*. Feeds sind in der Regel bidirektional, müssen es aber nicht sein. Läßt sich beim Newsfluß zwischen zwei Systemen ein deutliches Gefälle ausmachen, spricht man auch gelegentlich von *upstream* und *downstream*.

Die wenigsten Systeme haben heute noch einen vollen Feed, d.h. mit allen erhältlichen Newsgruppen; bei einem täglichen Volumen von bald 200 Megabytes allein in den sieben großen Hierarchien ist das für die wenigsten noch verkraftbar. Stattdessen beschränken sie sich auf diejenigen Gruppen oder Hierarchien, die für sie oder die von ihnen abhängigen Systeme interessant sind.

Je nach Art des zugrundeliegenden Netzes werden News auf verschiedene Arten transportiert. In UUCP-Netzen werden Artikel meist über den Zeitraum einer halben Stunde oder Stunde gesammelt, anschließend in große Dateien (sogenannte Batches) verpackt und komprimiert. Die Batches werden dann an das Programm *mnews* auf dem Nachbarsystem ausgeliefert.

Im Internet steht mit NNTP, dem Network News Transfer Protocol, ein wesentlich flexibleres Transportprotokoll zur Verfügung. NNTP transportiert im Gegensatz zu UUCP jeden Artikel einzeln und überprüft für jeden Artikel *vor* der Übertragung, ob das Zielsystem ihn vielleicht schon hat. Dabei übermittelt das sendende System die Message-ID des Artikels, das Empfängersystem prüft, ob es den Artikel bereits anderweitig empfangen hat. Wenn nicht, bittet es um dessen Übertragung. Wegen der dabei benutzten Protokoll-Meldungen heißt das Verfahren auch *ihave/sendme*.

## News-Software

Unter UNIX-Systemen gibt es traditionell zwei Kategorien von News-Software. Auf der einen Seite steht die interne Verwaltung von News, die ein- und ausgehende Artikel verwaltet, lokal auf der Platte ablegt und nach einiger Zeit wieder löscht. Hierfür existieren eine ganze Reihe von Paketen. Die bekanntesten sind C News und INN. C News wurde von Geoff Collyer und Henry Spencer geschrieben; die erste Version stammt aus dem Jahr 1987, wurde aber ständig weiterentwickelt. Die neueste Version ist das sogenannte Cleanup Release vom September 1994. INN steht für Internet News und wurde von Rich Salz entwickelt; die erste Version wurde 1992 veröffentlicht. Die aktuelle Version ist 1.4sec.

Auf der anderen Seite stehen die Newsreader, mit denen die Benutzer News lesen (daher der Name) und posten können. Einige Programme unterscheiden sich recht deutlich in der Bedienung, so daß sich das Probieren wirklich lohnt. Es hängt ganz von den persönlichen Vorlieben ab, welchen Newsreader jemand als ``den komfortabelsten'' bezeichnen würde. ☒ Der unter Linux am häufigsten benutzte Newsreader dürfte *tin* von Iain Lea sein; beliebt ist aber auch *nn* von Kim Storm.

Während es durchaus üblich ist, mehrere Newsreader zu installieren, um unterschiedliche Geschmäcker zufriedenzustellen, können Sie nur ein Newssystem benutzen. Im Rest dieses Abschnitts werden wir uns auf das im Vergleich zu C News wesentlich flexiblere und schnellere INN konzentrieren. Es wurde für den Einsatz in einer NNTP-Umgebung optimiert, unterstützt aber auch UUCP ohne Probleme.

---

[Next](#) [Up](#) [Previous](#) [Contents](#) [Index](#)

**Next:** [INN](#) **Up:** [Datenreisen und reisende Daten](#) **Previous:** [Elektronische Post mit smail](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

## Subsections


- [INN und IP-Networking](#)
  - [Administrativer Kleinkram](#)
  - [Globale Parameter](#)
  - [Die Dateien `active` und `newsgroups`](#)
  - [Wer bekommt was - `newsfeeds`](#)
  - [Leben und Sterben des `innd`](#)
  - [Ein ausgehender Newsfeed über UUCP](#)
  - [Löschen von Artikeln mit `expire`](#)
  - [Control-Messages](#)
  - [Overview-Dateien](#)
  - [Und jetzt ohne Hände...](#)
- 

# INN

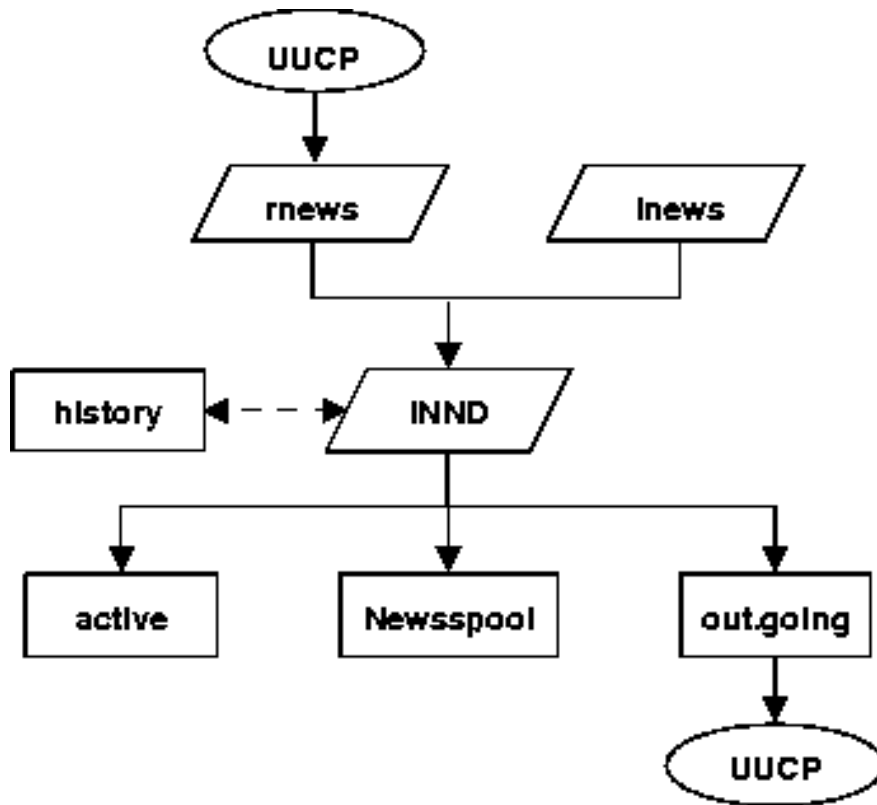
INN besteht aus einer ganzen Reihe von Dienstprogrammen und einer Art Chef-Programm, dem Dämon *innd*. Diese Programme befinden sich zusammen mit diversen Hilfsdateien im Verzeichnis `/usr/lib/news`. Außerdem gibt es noch das Verzeichnis `/var/spool/news`, kurz Newsspool genannt. Hier werden unter anderem die einzelnen Artikel abgelegt.

*innd* wird beim Booten des Rechners gestartet und läuft von da an im Hintergrund, bis Sie den Rechner wieder herunterfahren. Diese Methode beschleunigt die Bearbeitung von News ganz erheblich, weil auf diese Weise die Statusdateien nicht für jeden ankommenden Newsbatch neu gelesen werden müssen, sondern nur einmal beim Start des Dämons. Zu den Aufgaben von *innd* gehört es, alle hereinkommenden Artikel zu bearbeiten, lokal abzuspeichern und gegebenenfalls an andere Programme weiterzugeben, z.B. zur Archivierung oder zum Transport an Nachbarsysteme.

Andere wichtige Komponenten von INN sind *rnews*, *inews* und *nnrpd*. Wenn ein anderes System per UUCP einen Newsbatch an Sie schickt, wird der bei der Ankunft auf Ihrem System von *rnews* in Empfang genommen, der die einzelnen darin enthaltenen Artikel an *innd* übergibt. *inews* ist das Pendant dazu: wenn Sie von Ihrem Newsreader aus einen Artikel posten, vervollständigt er die Header-Informationen und reicht ihn dann an *innd* weiter. *nnrpd* schließlich bedient Newsreader, die über NNTP auf das Newssystem zugreifen. Direkte NNTP-basierte Feeds werden von *innd* selbst bedient. Die einzelnen Komponenten kommunizieren über Netzwerk-Sockets miteinander.

Abbildung [7.12](#) zeigt schematisch den Newsfluß durch INN. In der oberen Hälfte des Bildes sehen Sie die Einlieferung von News über *rnews* und *inews* angedeutet. Die Ellipse soll eine Netzverbindung über UUCP symbolisieren.  In der unteren Hälfte ist dargestellt, auf welchen Wegen Informationen von *innd* weitergegeben werden; wir werden uns mit diesen Komponenten gleich eingehender beschäftigen.


**Abbildung:** Newsfluß durch INN




Wenn *innd* einen Artikel empfängt, prüft er zunächst anhand der Message-ID, ob er ihn schon einmal bearbeitet hat oder nicht. Falls sich die ID bereits in der Datei `/usr/lib/news/history` findet, verwirft er sie einfach. Handelt es sich aber wirklich um einen neuen Artikel, legt *innd* ihn lokal ab und stellt anhand der Datei `/usr/lib/news/newsfeeds` fest, ob er ihn außerdem an andere Sites weitergeben, ihn archivieren und eventuell noch ganz andere Dinge mit ihm anstellen soll.

Artikel werden unterhalb des Verzeichnisses `/var/spool/news` gespeichert (dem Newsspool, wie schon erwähnt). Jeder Newsgruppe entspricht ein Verzeichnis, in dem jeder Artikel in einer separaten Datei abgelegt wird. Die Dateinamen sind fortlaufende Nummern, so daß ein Artikel in `comp.risks` beispielsweise in `comp/risks/217` abgelegt wird. Die jeweils letzte Artikelnummer einer Gruppe vermerkt INN in der Datei `/usr/lib/news/active`.

Soll ein Artikel über UUCP an ein anderes System weitergegeben werden, sagen wir *cicero*, legt INN den Dateinamen in der Datei `outgoing/cicero` im Newsspool ab. Diese Datei wird später vom Batchskript *send-uucp* gelesen, das die Artikel in einen oder mehrere Batches verpackt und über UUCP an *cicero* weiterschickt.

Nach soviel Theorie werden wir jetzt etwas konkreter und wenden uns der eigentlichen Installation zu. Wir beschränken uns hier auf eine ganz einfache Konfiguration eines UUCP-Systems mit nur einem Newsfeed. INN kann natürlich noch eine ganze Menge mehr, aber eine ausführliche Beschreibung all dieser Möglichkeiten würde wohl den Rahmen dieses Buches sprengen. Falls Sie mehr über INN wissen wollen, empfehlen wir Ihnen die (englischsprachige) Dokumentation, die zur INN-Distribution dazugehört, sowie die monatlich in `news.answers` und `news.software.b` gepostete INN-FAQ. 

# INN und IP-Networking

Eine der Schwierigkeiten, die Anfänger manchmal mit INN haben, ist die Tatsache, daß die einzelnen Komponenten von INN über Netzwerkverbindungen (Sockets) kommunizieren - auch, wenn sich alles nur auf einer einzelnen Maschine abspielt.  Sie brauchen also für den Betrieb von INN einen Kernel mit zumindest minimalen Netzwerkfähigkeiten. In diesem Kapitel werden Sie deshalb einem Crash-Kurs in Sachen Netzwerk-Konfiguration unterzogen. Aus Platzgründen kann ich nur beschreiben, *wie* Sie das tun - für Erklärungen reicht's leider nicht aus. Wenn Sie sich für diese Details interessieren, lege ich Ihnen (nicht ohne rot zu werden) den *Linux Network Administrator's Guide* ans Herz.

Wenn Sie Ihren Rechner bereits für eine Art von Networking konfiguriert haben, brauchen Sie sich um die meisten Dinge in diesem Abschnitt nicht zu kümmern. Die einzige Ausnahme gilt für Maschinen, deren einzige Netzverbindung ein SLIP- oder PPP-Link ist; sie sollten unbedingt das Dummy-Device konfigurieren, wie im folgenden beschrieben. Wenn Sie bisher noch überhaupt keine Netzwerksoftware installiert haben, sollten Sie das jetzt nachholen.

Anschließend müssen Sie den Kernel neu übersetzen, wie in [Kapitel](#) beschrieben. Bei der Konfiguration müssen Sie die Frage nach TCP/IP Networking mit `y` beantworten. Bei den Netzwerktreibern sollten Sie das Dummy-Device einschalten, wenn Sie keine permanente IP-Verbindung zur Außenwelt haben. Das Dummy-Device ist eine Art ``Haken'', an dem Sie eine IP-Adresse festmachen können.

Anschließend müssen einige Konfigurationsdateien angepaßt werden. Achten Sie darauf, daß in der Datei `/etc/hosts` der volle Hostname Ihres Rechners (mit Domain) eingetragen ist:

```
# Beispiel-Datei /etc/hosts
# IP-Adresse      Name(n)
127.0.0.1         localhost
192.168.1.1       isis.in-berlin.de isis
```

Die seltsamen Nummern in der linken Spalte sind IP-Adressen; sie werden vom Netzwerk benutzt, um Rechner weltweit eindeutig zu identifizieren. Wenn Sie nicht netzwerktechnisch bereits ans Internet angebunden sind, werden Sie sicherlich keine solche Adresse haben. Woher also nehmen wenn nicht stehlen? Das ist nicht weiter schwierig; die Adresse `127.0.0.1` müssen Sie sowieso für den Namen *localhost* verwenden (das gehört so). Und wenn Sie sowieso nicht ans Internet angeschlossen sind, muß die Adresse Ihrer Maschine auch nicht mehr eindeutig sein; Sie können also getrost die aus dem Beispiel übernehmen.



Als nächstes editieren Sie die Datei `/etc/rc.d/rc.inet1`; hier muß sinngemäß folgendes stehen:

```
# Beispiel-Datei /etc/rc.d/rc.inet1.
# Konfiguration des Loopback-Device
/sbin/ifconfig lo localhost
/sbin/route add -net 127.0.0.0

# Konfiguration des Dummy-Device
/sbin/ifconfig dummy isis
/sbin/route add isis
```

Wenn Sie Slackware benutzen, müssen Sie außerdem noch die Dateien `NETWORKING` und `HOSTNAME` im Verzeichnis `/etc` anpassen; die erste muß einfach den String `yes` enthalten; die andere den vollen Hostnamen (in unserem Beispiel also *isis.in-berlin.de*).



# Administrativer Kleinkram

Damit INN auf Ihrem System problemlos laufen kann, müssen zunächst einige Vorbedingungen erfüllt sein.

Als erstes müssen auf Ihrem System ein Benutzer und eine Gruppe namens `news` existieren. Alle Dateien und Programme, die Teil von INN sind, müssen diesem Benutzer und zu dieser Gruppe gehören. Das ist sehr wichtig; sollten Sie einmal aus Versehen eine Datei einem anderen Benutzer zuordnen, kann sie dadurch für INN unzugänglich werden und so Ihr gesamtes Newssystem lahmlegen. Aus diesem Grunde sollten Sie alle Operationen, die INN betreffen, als `news` ausführen, wenn nicht anders beschrieben. Ausnahmen sind der Start des Newssystems über `rc.news` (dieses Skript *muß* als `root` laufen) und `ctlinnd` (dieses Programm *dürfen* Sie auch als `root` aufrufen). Beide Programme werden in späteren Abschnitten noch ausführlicher besprochen.

Im folgenden werden Befehle, die Sie als `root` ausführen können, durch `root#` gekennzeichnet; solche, die Sie als `news` aufrufen müssen, werden durch den Shell-Prompt `news$` gekennzeichnet. Für letztere ist es nicht unbedingt nötig, sich auch tatsächlich als `news` einzuloggen. Folgende Methode ist recht bequem, um ein Kommando als `news` aufzurufen:

```
root# su news -c "kommando"
root# _
```

Verschiedene zu INN gehörige Programme schicken dann und wann eine Mail an die News-Administratorin, beispielsweise, um auf ein Problem hinzuweisen. Diese Nachrichten werden an den Benutzer `newsmaster` geschickt. Sie sollten für diese Adresse einen Alias in der Datei `/usr/lib/aliases` eintragen, der auf die für die Wartung von INN zuständige Person zeigt:


```
# newsmaster-Alias in /usr/lib/aliases
newsmaster:      karla
```

Außerdem sollten Sie darauf achten, daß die Programme *inews* und *rnews* ``im Pfad" stehen, das heißt, daß Benutzer sie aufrufen können, ohne den vollen Pfadnamen angeben zu müssen. Üblicherweise legt man dafür symbolische Links von `/usr/bin` oder `/usr/local/bin` an, die auf die tatsächlichen Programme verweisen. Wenn Ihr INN aus einer Linux-Distribution stammt, sollte das eigentlich von der Installationsprozedur erledigt werden. Das ist aber nicht immer der Fall. Prüfen Sie deshalb nach, ob diese Links existieren; wenn nicht, legen Sie sie mit den folgenden Befehlen an:

```
root# cd /usr/bin
root# ln -s ../lib/news/rnews rnews
root# ln -s ../lib/news/inews inews
root# _
```

Schließlich sollten Sie darauf achten, daß die für den Betrieb von INN nötigen Systemressourcen vorhanden sind. INN benötigt im laufenden Betrieb je nach Größe Ihres Newsfeeds mindestens 1.5 Megabyte Swap-space für den Dämon *innd* und eventuell zugehörige Prozesse wie *overchan*. Die allermeisten Sites werden mit diesen 1.5 MB auskommen; auf großen Systemen kann *innd* aber auch wesentlich ``fetter" werden und 8 MB oder mehr benötigen.

Außerdem benötigen Sie einiges an Plattenplatz. Ein wesentlicher Faktor ist der Newsspool. Wenn Sie nur einen kleinen Feed von ein paar Dutzend Newsgruppen durchschnittlicher Größe haben und die meisten Artikel nach wenigen Tagen wieder weggeworfen werden, können Sie durchaus mit 20 MB auskommen.

Der Normalfall wird aber eher bei ca. 40 MB und darüber liegen.  Zusätzlich benötigen Sie auf der Platte, auf der sich das Verzeichnis `/usr/lib/news` befindet, neben dem Platz für die Programme,


Skripte und so weiter (ca. 2.5 MB) noch mindestens 1 MB für die Datei `history`.


# Globale Parameter

Die Haupt- und Staatskonfiguration des INN findet in der Datei `/usr/lib/news/inn.conf` statt. Hier wird unter anderem der Hostname festgelegt, unter dem Ihr Rechner im Usenet firmiert. Das ist in den meisten Fällen Ihr voller Domainname; vereinzelt wird hier aber auch noch der UUCP-Name verwendet.

```
# inn.conf - Beispiel fuer isis.in-berlin.de
#
server:          isis.in-berlin.de
domain:         in-berlin.de
moderatormailer: %s@uunet.uu.net
organization:    Lives in Limbo
```

Die erste Zeile dieser Beispieldatei legt fest, mit welchem Host die Programme *rnews* und *inews* Kontakt aufnehmen, wenn sie einen Artikel oder Batch an *innd* abliefern wollen. In unserem Fall bleibt ihnen keine andere Wahl als *isis* selbst.

Das zweite Attribut legt den Namen der Domain fest, in der der Rechner sich befindet. Dieses Attribut müssen Sie nicht unbedingt angeben, da *innd* meist in der Lage ist, den vollen Systemnamen aus `/etc/hosts` zu erfragen. 

Die nächste Zeile gibt eine Adresse an, an die Artikel für moderierte Newsgruppen geschickt werden sollen. Ein Artikel, den Sie in die moderierte Gruppe `soc.feminism` posten, wird dann von INN automatisch an `soc-feminism@uunet.uu.net` geschickt. Auf UUNET gibt es für jede moderierte Usenet-Gruppe einen passenden Alias, der Ihren Artikel dann an die richtige Person weiterleitet. 

Das Attribut `organization` legt den Text fest, den INN in das `Organization:`-Feld jedes Artikel-Headers einfügt, der auf Ihrer Maschine gepostet wurde. Wenn Sie nicht gerade einen Ruf zu verlieren haben, gilt es durchaus als chic, hier dem eigenen Sinn für Humor freien Lauf zu lassen.

Neben `inn.conf` müssen Sie auch noch die Dateien `hosts.nntp` und `nnrp.access` für Ihr System anpassen, die den Zugang zu Ihrem Newssystem festlegen. `hosts.nntp` regelt, welche Systeme News über *rnews* und direkte NNTP-Verbindungen einliefern dürfen, während `nnrp.access` für das Posten von Artikeln über *inews* und NNTP-basierte Newsreader zuständig ist.

Auf einem System wie *isis* ist es nicht sonderlich interessant, den Zugriff auf INN einzuschränken; diese Mechanismen entfalten ihre volle Wirkung erst auf einer Internet-Site. Um aber selbst News posten zu können, müssen Sie zumindest Ihre eigene Maschine in diesen Dateien eintragen, weil sich INN sonst schlichtweg weigert, Artikel von *irgendwem* anzunehmen.

Die Konfiguration von `hosts.nntp` für *isis* sieht so aus:

```
# /usr/lib/news/hosts.nntp fuer isis
isis.in-berlin.de:
```

Fast genauso simpel ist `nnrp.access`:

```
# /usr/lib/news/nnrp.access fuer isis
isis.in-berlin.de:Read Post:::*
```



Ich will hier nicht auf die Bedeutung der einzelnen Felder eingehen; wenn Sie sich mit diesen Mechanismen beschäftigen wollen, finden Sie die nötigen Informationen in den Manual-Seiten `nntp.access(5)` und `hosts.nntp(5)`.

## Die Dateien `active` und `newsgroups`

Über die `active`-Datei findet die Buchhaltung des Newssystems statt. Sie enthält Informationen über den aktuellen Status jeder einzelnen Newsgruppe. Ein Ausschnitt aus einer `active`-Datei stellt sich beispielsweise so dar:

```
de.comp.databases 0000000813 0000000803 y
de.comp.dtp 0000000586 0000000577 y
de.comp.gnu 0000001469 0000001462 y
de.comp.graphik 0000001121 0000001108 y
de.comp.lang.c 0000002398 0000002334 y
de.comp.lang.c++ 0000001790 0000001761 y
de.comp.lang.forth 0000000270 0000000268 y
de.comp.lang.lisp 0000000039 0000000040 y
de.comp.lang.misc 0000000374 0000000358 y
de.comp.lang.pascal 0000001824 0000001776 y
```

Jede Zeile beginnt mit dem Namen der Newsgruppe. Die folgenden zwei Zahlen geben jeweils die höchste und niedrigste Nummer der aktiven Artikel in dieser Gruppe an. Enthält die Gruppe im Augenblick keine Artikel, ist die zweite Zahl um eins höher als die erste. Das letzte Feld enthält ein Flag, das den Status der Gruppe festlegt. Am häufigsten wird Ihnen dabei `y` für ``yes" begegnen, was bedeutet, daß lokale Postings in diese Gruppe erlaubt sind. `n` verbietet das, läßt aber immer noch zu, daß die hereinkommenden Artikel von anderen Systemen im Newsspool abgelegt werden, und `m` bezeichnet eine moderierte Gruppe. Daneben gibt es noch die Flags `x`, `j` und `=`, die aber so gut wie nie benötigt werden. Die Bedeutung dieser Flags finden Sie in der Manpage `active(5)` erklärt.

`inn` liest diese Datei beim Start ein und behält sie während der gesamten Laufzeit teilweise im Speicher. Sie sollten diese Datei also niemals editieren, während `INN` läuft! Die Folge ist meistens ein heilloses Durcheinander, das sich nur unter größten Verrenkungen beheben läßt.

Bei der Erstkonfiguration Ihres Systems sollten Sie sich eine aktuelle `active`-Datei von Ihrem Newsfeed besorgen. Sie können sie getrost unverändert installieren; wenn Sie wollen, können Sie aber auch vorher noch die Artikelnummern zurücksetzen. Das geht beispielsweise mit den folgenden Befehlen:

```
root# cd /usr/lib/news
root# sed 's/ [0-9]* [0-9]* / 000000000 000000001 /' active > active.new
root# mv active.new active
root# chown news.news active
root# chmod 644 active
root# _
```


Wenn Sie die `active`-Datei installieren, sollten Sie sicherstellen, daß es die Newsgruppen `junk` und `control` enthält. `junk` ist so etwas wie der Mülleimer Ihres Newssystems. Ein Artikel wird dann in `junk` abgelegt, wenn keine der Gruppen, in die er gepostet wurde, in Ihrer `active`-Datei vorkommt. Sie sollten deshalb ab und zu in `junk` hineinschauen; wenn die Gruppe nicht leer ist, sollten sie die fehlenden Gruppen entweder bei Ihrem Feed abbestellen oder sie lokal einrichten (wie, wird in Abschnitt [7.7.9](#)

erklärt).

Die Gruppe `control` dient als Ablage der sogenannten Control-Messages, die ebenfalls in Abschnitt [7.7.9](#) behandelt werden.

Zu guter Letzt sollten Sie sich ein aktuelles Exemplar der Datei `newsgroups` besorgen. Sie wird zwar von INN nicht gebraucht, ist aber trotzdem für Ihren Newsreader ganz nützlich. Diese Datei enthält nämlich zu jeder Gruppe eine Kurzbeschreibung in wenigen Worten. Viele Newsreader zeigen diesen Text in der Übersicht der Newsgruppen an, so daß Sie sich schneller orientieren können.

## Wer bekommt was - newsfeeds

Die Datei `newsfeeds` stellt den zentralen Verteiler von INN dar. Hier wird vor allem festgelegt, welche Newsgruppen an welche Sites weitergegeben werden. Beachten Sie, daß `newsfeeds` nichts darüber aussagt, was für Gruppen Ihr Feed an Sie ausliefert - das wird ausschließlich in dessen `newsfeeds`-Datei festgelegt. 

Eine Beispielkonfiguration für *isis* sieht so aus:


```
# Beispiel-Konfiguration newsfeeds fuer isis.in-berlin.de
#
ME\
    :!* /!local\
    ::

# cicero.lunetix.de - unser groesster und einziger Newsfeed
cicero/cicero.lunetix.de,news.lunetix.de\
    :*,!junk,!local /!local\
    :Tf,Wfb:
```

Einträge in dieser Datei bestehen aus vier Feldern, getrennt durch Doppelpunkte. Das erste Feld enthält den Namen des Feeds, z.B. *cicero*. Dem Namen folgt optional eine Ausschlußliste, abgetrennt durch einen Schrägstrich. Bevor *innd* einen Artikel an einen Feed weitergibt, überprüft er anhand der `Path:`-Zeile im Header, ob der Artikel bereits über diese Site gelaufen ist. Dabei bezieht er nicht nur den Namen des Feeds (hier also *cicero*), sondern auch die der Sites in die Ausschlußliste (d.h. *cicero.lunetix.de* und *news.lunetix.de*) mit ein.

Das zweite Feld enthält die Liste aller Newsgruppen und Hierarchien, die an den Feed weitergereicht werden. Da es etwas unhandlich wäre, alle Newsgruppen einzeln aufzuführen, werden hier bestimmte Ausdrücke, sogenannte wildmat-Patterns, angegeben. Diese Ausdrücke funktionieren so ähnlich wie die Wildcards in der Shell, d.h. `comp.*` trifft auf alle Untergruppen der Hierarchie `comp` zu, und `comp.os.v*` auf Gruppen wie `comp.os.v` und `comp.os.vms`. In `newsfeeds` können Sie mehrere dieser Patterns in einer Liste kombinieren, wie beispielsweise `de.*,!de.comp.sys.*`. Dieser Ausdruck wählt alle Gruppen in der Hierarchie `de` mit Ausnahme der Gruppen unterhalb von `de.comp.sys` aus.


Zur Liste der Newsgruppen gehört auch noch eine Liste von gültigen Distributionen; das ist der Teil nach dem Schrägstrich. Sie können nämlich im Artikel-Header eine `Distribution:`-Zeile angeben, die die Verteilung des Artikels weiter einschränken soll. So gab es eine Zeitlang eine Distribution `de`, die sicherstellen sollte, daß Artikel den deutschsprachigen Teil des Usenet nicht verlassen. Diese

Distributionen sind aus verschiedenen Gründen aus der Mode gekommen.  Die einzige Distribution, der allgemein noch eine Daseinsberechtigung zugestanden wird, ist `local`.

Schauen wir uns jetzt unser Beispiel an, so werden diese kryptischen Zeilen etwas klarer: wir geben an *cicero* alle Newsgruppen außer von `junk` und `local` weiter (das ist der Teil `*, !junk, !local`). Eine Ausnahme sind Artikel mit der Distribution `local`, sie werden auf gar keinen Fall weitergereicht (das ist der Teil mit `!local`).

Das dritte Feld eines Eintrags in `newsfeeds` enthält diverse Flags, die INN mitteilen, um was für eine Art Feed es sich handelt. Es ginge hier zu weit, die Bedeutung dieser Flags im einzelnen zu erklären; Details finden Sie in der Manpage `newsfeeds(5)`.

Im Falle von *cicero* besagen die Flags `Tf`, `Wfb`, daß INN die Pfadnamen der zu sendenden Artikel in einer Datei ablegen soll. Der voreingestellte Standardname für die Datei ist `/var/spool/news/out.going/cicero`. Sie können aber im letzten Feld des Eintrags einen anderen Namen angeben, wenn Sie wollen. Uns ist der Default aber ganz recht, da wir die Liste der Artikel später mit dem Batcher bearbeiten wollen; dieses Programm erwartet die Datei eben in `out.going`. Das `b` in `Wfb` bewirkt, daß *innd* neben dem Pfadnamen auch noch die Länge des Artikels in Bytes ablegt; dadurch kann der Batcher die Größe der ausgehenden Batches genauer berechnen.

Jetzt sind wir so lange auf dem *cicero*-Eintrag herumgeritten; was hat es denn nun mit ME auf sich? Der Eintrag ME spielt eine besondere Rolle, er enthält die Default-Liste der Newsgruppen, die an andere Sites weitergegeben werden dürfen. Angenommen, Sie haben lokal die Newsgruppe `secret` eingerichtet, in denen Sie mit Ihren Kollegen über das Rezept für den perfekten Taco diskutieren; dann brauchen Sie nur `*, !secret` ins Newsgruppenfeld von ME einzutragen, um zu verhindern, daß diese Gruppe weiterverbreitet wird. Ansonsten hat der ME-Eintrag (was ehemalige C News-Benutzer vielleicht etwas überraschen wird) keine weitere Bedeutung. 

## Leben und Sterben des *innd*

Wie bereits erwähnt, wird *innd* beim Booten des Systems gestartet. Das geschieht durch das Skript *rc.news*. In den meisten Linux-Distributionen wird das Skript nicht automatisch ausgeführt, sondern Sie müssen diesen Aufruf noch von Hand in eines der *rc*-Skripte einfügen. Ein günstiger Platz dafür ist in `/etc/rc.d/rc.local`, zum Beispiel durch den folgenden Befehl:


```
# start INN
if [ -x /usr/lib/news/etc/rc.news ]; then
    /bin/sh /usr/lib/news/etc/rc.news
fi
```

Da wir vom Booten reden, paßt es ganz gut, uns auch gleich mit dem umgekehrten Vorgang zu befassen. Beim Herunterfahren des Systems schickt das *shutdown*-Kommando allen Prozessen das TERM-Signal. Der unschöne Nebeneffekt ist, daß *syslogd* meist vor *innd* die Arbeit einstellt, so daß *innd* eine ganze Reihe bedrohlich wirkender Meldungen auf den Bildschirm ausgibt. Wenn Sie das vermeiden wollen, können sie INN vorher ordnungsgemäß herunterfahren. Das geschieht durch das folgende Befehlspaar:

```
root# /usr/lib/news/bin/ctlinnd throttle "system shutdown"
root# /usr/lib/news/bin/ctlinnd shutdown "system shutdown"
root# _
```

Der Befehl *ctlinnd* ist sozusagen Ihr Sprachrohr, wenn Sie sich zur Laufzeit mit Ihrem *innd* unterhalten wollen. Die Wirkung des Befehls `shutdown` brauche ich wohl nicht weiter auszumalen. Interessant ist

aber das Kommando `throttle` - es ``drosselt" den Server. In diesem Zustand akzeptiert *innd* keine weiteren NNTP-Verbindungen mehr und bearbeitet auch die bestehenden Verbindungen nicht. Damit können auch *rnews* und *inews* keine Artikel mehr einliefern.

*ctlinnd* ist aber nicht nur zum Herunterfahren des Servers nützlich. Sollte es einmal nötig werden, eine Datei wie `/usr/lib/news/history` zu editieren,  müssen Sie dazu nicht extra das Newssystem herunterfahren. Stattdessen können Sie folgende Befehle eingeben:

```
root# cd /usr/lib/news
root# bin/ctlinnd throttle "edit history"
root# vi history
....
root# bin/ctlinnd reload history "fixed corruption"
root# bin/ctlinnd go "edit history"
root# _
```


Der `go`-Befehl hebt eine vorhergehende Drosselung des Servers wieder auf.

Die Strings, die hier bei jedem Befehl mit angegeben werden, werden von *innd* an *syslogd* übergeben; sie sollten den Grund für die Drosselung usw. bündig benennen. Der genaue Wortlaut des Texts ist aber völlig nebensächlich. Die einzige Bedingung ist, daß Sie beim `go`-Kommando denselben Grund angeben wie beim vorhergehenden `throttle`.

Das Kommando `reload` veranlaßt *innd*, die angegebene Datei neu zu laden. Neben `history` sind noch Parameter wie `active`, `newsgroups` oder `all` gültig.

## Ein ausgehender Newsfeed über UUCP


Wir haben im vorletzten Abschnitt bereits gesehen, was Sie in `newsfeeds` eintragen müssen, um Artikel an ein anderes System weitergeben zu können. Das ist aber nur die halbe Geschichte, denn *innd* tut zunächst nichts anderes, als die Pfadnamen und Größe der Artikel in einer Datei einzutragen - für *cicero* war das `out.going/cicero` im Newsspool. Sie muß zu einem späteren Zeitpunkt vom Batcher weiterverarbeitet werden, der die genannten Artikel in einen oder mehrere Batches verpackt und auf die Reise zu *cicero* schickt.

Bei INN ist das zentrale Stück dieses Mechanismus das Programm *batcher*. Es wird aber meist nicht direkt aufgerufen, sondern über Shell-Skripte, die noch einige Magie drumherum veranstalten.  Ein solches Skript ist *send-uucp*, dem Sie beim Aufruf den Namen des Systems übergeben, für das es die Päckchen schnüren soll:

```
news$ send-uucp cicero
news$ _
```

Auch hier ist es wichtig, daß *send-uucp* nur unter der Benutzer-ID `news` läuft. Am sinnvollsten ist es, *send-uucp* in regelmäßigen Abständen von *cron* aus aufzurufen. Wir zeigen in Abschnitt [7.7.11](#) ein Beispiel hierfür.

*send-uucp* greift sich die Datei `out.going/cicero` und verfüttert sie an *batcher*, der daraus einen komprimierten Newsbatch erzeugt. Die fertigen Batches werden mittels UUCP an das Kommando *rnews* auf *cicero* übermittelt. UUCP überträgt die Daten aber nicht umgehend, sondern spoolt sie nur. Die tatsächliche Übertragung findet erst dann statt, wenn Sie das nächste Mal die Verbindung zu *cicero* aufbauen.

*send-uucp* ist leider nicht optimal und bietet bei weitem nicht den Komfort, den ehemalige C News-Benutzer von ihrem *sendbatches* gewöhnt sind. Zu seinen Schwächen gehört, daß Sie Parameter wie die maximale Größe der Batches nur für alle Nachbarsysteme gleich einstellen können, und das auch nur, wenn Sie das Skript editieren. Es ist also wirklich nicht für viel mehr geeignet, als auf einer kleinen Leafsite  die lokal erzeugten Artikel ``stromaufwärts" zu seinem Newsfeed zu transportieren. Für größere Systeme empfiehlt es sich, per FTP das *sendbatches*-Paket von *grasp.univ-lyon1.fr* zu besorgen.

Hier ist der Ort, darauf hinzuweisen, daß in manchen Linux-Distributionen das Programm *compress* ein symbolischer Link auf *gzip* ist. Traditionell wird zum Komprimieren der Batches *compress* benutzt, dessen Komprimierung mit der von *gzip* nicht kompatibel ist. Das kann bei der Kommunikation mit Newssystemen, die kein Linux fahren, Probleme bereiten. Die korrekte Lösung des Problems ist hier, den symbolischen Link zu löschen und stattdessen das richtige *compress* zu installieren.

## Löschen von Artikeln mit *expire*

Wenn Sie vermeiden wollen, daß Ihre Festplatte nach kürzester Zeit überläuft, müssen Sie regelmäßig alte Artikel löschen. Dafür ist bei INN das Programm *expire* zuständig. Sie sollten es mindestens einmal täglich laufen lassen, bei höherem Aufkommen auch öfter. Eine mögliche Art, es aufzurufen, ist diese:

```
news$ cd /usr/lib/news
news$ bin/ctlinnd throttle "running expire"
news$ bin/expire -s -v1
news$ bin/ctlinnd go "running expire"
news$ _
```

Achten Sie darauf, daß Sie *expire* nur als Benutzer *news* aufrufen. Wenn Sie das als *root* tun, können unter Umständen die Zugriffsrechte für einige Dateien so verändert werden, daß andere Teile von INN nicht mehr darauf zugreifen können, was für alle Seiten fatale Folgen hat.

Dieser Aufruf löscht alle ``alten" Artikel aus Ihrem Newsspool (wir werden gleich sehen, wie alt ``alt" ist). Die Flags *-s* und *-v1* erzeugen zusätzliche Ausgaben über die Anzahl der gelöschten Artikel, den derzeit verbrauchten Plattenplatz usw. Eine ganze Reihe weiterer Flags finden Sie in der Manual-Seite *expire(8)* erklärt.

Die Arbeit von *expire* wird über die Datei *expire.ctl* gesteuert. Hier können Sie für einzelne Newsgruppen oder -hierarchien angeben, nach wievielen Tagen Artikel in diesen Gruppen weggeworfen werden. Sie sollten mit diesen Werten ruhig experimentieren; neben dem vorhandenen oder nicht vorhandenen Plattenplatz ist es vor allem eine Frage der persönlichen Vorlieben, welche Newsgruppen Sie länger vorhalten und welche nicht.

Auf *isis* könnte *expire.ctl* so aussehen:

```
## expire.ctl
## So lange wird die Message-ID in der history aufgehoben:
/remember/:14

## Default: 2 Tage
*:A:1:2:2

## junk und control verschwinden umgehend
```



```
junk:A:1:1:2
control:A:1:1:2
```


```
## die groesseren Hierarchien
sci.*,comp.*:A:1:2:14
de.*:A:1:4:30
alt.*:A:1:1:30
```

```
## comp.os.linux.* bleibt etwas laenger
comp.os.linux*,comp.unix.*:U:1:7:31
comp.os.linux*,comp.unix.*:M:1:14:31
```

Im ersten Feld jeder Zeile begegnen uns die wildmat-Patterns wieder, die Sie bereits in der Datei `newsfeeds` gesehen haben. Sie legen fest, auf welche Gruppen der jeweilige Eintrag zutreffen soll, der Ausdruck `sci.* , comp.*` betrifft beispielsweise alle Newsgruppen der Hierarchien `sci` und `comp`. `expire` "durchsucht" die Datei sequentiell und verwendet dabei für eine bestimmte Gruppe jeweils die letzte passende Zeile. Ein Artikel in `comp.os.linux.announce` würde von der letzten Zeile gematcht, obwohl einige Zeilen weiter oben bereits ein Eintrag für die ganze Hierarchie `comp` auftaucht.

Auf den wildmat-Ausdruck folgen vier weitere Felder, die durch Doppelpunkte voneinander getrennt sind. Das wichtigste ist das vorletzte, das angibt, wieviele Tage ein Artikel aus dieser Gruppe im Normalfall vorgehalten wird. Es ist wichtig zu wissen, daß ab dem Zeitpunkt des Eintreffens auf Ihrem System gerechnet wird, nicht ab dem Zeitpunkt, als der Artikel gepostet wurde.

Das dritte und fünfte Feld legen die Mindest- und Höchstdauer fest, die ein Artikel in Ihrem Newsspool bleiben darf. Das erscheint auf den ersten Blick widersinnig - wenn alle Artikel nach fünf Tagen weggeworfen werden, wie soll dann nach einem Monat immer noch einer übrig sein?!


Das hängt mit einem weiteren Feld im Artikel-Header zusammen, mit dem Sie die Lebensdauer eines Artikels künstlich hochsetzen können, der `Expires:-`Zeile. Bei manchen Artikeln ist es durchaus erwünscht, daß sie länger aufgehoben werden als gewöhnliche Artikel, zum Beispiel bei periodisch geposteten Informationen wie den Einführungen in `de.newusers` oder der Linux-FAQ. Diese Artikel erhalten vom Absender ein ausdrückliches Verfallsdatum in `Expires:` mit auf den Weg, das von Ihrem Newssystem ausgewertet wird. Ein solcher Artikel wird erst dann gelöscht, wenn entweder das festgesetzte Verfallsdatum erreicht ist, die Maximalzeit, die Sie ihm in `expirectl` zugestanden haben, abgelaufen ist, oder eine neuere Version der FAQ eingetroffen ist. 


Die minimale Lebenszeit eines Artikels, die in Feld drei enthalten ist, hat eine verwandte Funktion. Sie verhindert, daß ein Artikel sofort weggeworfen wird, wenn das Datum im `Expires`-Header schon abgelaufen wird, wenn er Ihr System erreicht. Das kommt selten genug vor, und es gibt kaum einen Grund, diesen Wert höher als einen Tag anzusetzen.

Jetzt können Sie einen Eintrag wie `de.*:A:1:4:30` fast vollständig entziffern: der Eintrag betrifft alle Gruppen unter `de` und wirft das Gros der Artikel nach 4, spätestens aber nach 30 Tagen fort. Jeder Artikel wird aber mindestens einen Tag vorgehalten.

Was noch fehlt, ist die Bedeutung des zweiten Feldes. Es enthält ein Flag, mit dem Sie noch einmal zwischen moderierten und unmoderierten Gruppen differenzieren können. Ein `M` schränkt den Eintrag auf moderierte Gruppen ein, ein `U` auf unmoderierte. Ein `A` ignoriert den Unterschied. In den letzten beiden Zeilen unseres Beispiels sehen Sie eine Anwendung dieser Flags.

```
comp.os.linux*,comp.unix.*:U:1:4:31
comp.os.linux*,comp.unix.*:M:1:14:31
```

Artikel in moderierten Untergruppen von `comp.os.linux` und `comp.unix`  werden hier erst nach zwei Wochen gelöscht, während solche aus unmoderierten Gruppen bereits nach 4 Tagen verschwinden.

Ein besonderer Eintrag in `expire.ct1` verdient noch unsere Beachtung; das ist die Zeile, die mit `/remember/` beginnt. Sie legt fest, wie lange sich INN eine Message-ID in der Datei `history` merkt. Würde INN nämlich die ID eines Artikels sofort wieder entfernen, sobald `expire` diesen gelöscht hat, könnte es passieren, daß ein Artikel ein zweites Mal akzeptiert wird, falls er noch einmal vorbeikommt. 

In unserem Beispiel ist der `remember`-Wert auf 14 Tage eingestellt. Das ist ganz sinnvoll für eine kleine Site wie *isis*, wo die meisten Duplikate (wenn sie mal auftreten) bereits von *cicero* abgefangen werden.

## Control-Messages

Es kommt gelegentlich vor, daß neue Newsgruppen eingerichtet oder alte gelöscht werden. Das geschieht im Usenet automatisch über sogenannte `newgroup` und `rmgroup` Control-Messages. Sie werden wie gewöhnliche Artikel durchs Usenet transportiert, unterscheiden sich aber durch eine zusätzliche Headerzeile namens `Control:`. Sie enthält den Typ der Control-Message und eventuelle Parameter. Für die Einrichtung von `de.comp.os.tunix` müßte der Artikel etwa folgende Zeile enthalten:

```
Control: newgroup de.comp.os.tunix
```

Neben `newgroup` und `rmgroup` gibt es noch einige andere Control-Messages, z.B. `cancel` zum nachträglichen Löschen eines Artikels durch den Absender, oder etwas exotischere wie `sendsys`, `senduuname` und `checkgroups`.

INN bearbeitet diese Nachrichten halbautomatisch. Das `Canceln` von Artikeln erledigt *innd* meist selbständig, während `newgroup` und `rmgroup` meist als Mail an die administrative Adresse `news` gesendet werden. Diese Mail enthält neben dem ursprünglichen Artikel auch den korrekten Aufruf von *ctlinnd*, mit dem Sie die Gruppe dann auch tatsächlich einrichten können. Beispielaufrufe zum Einrichten und Entfernen der Newsgruppe `local.test` sehen so aus:

```
root# ctlinnd newgroup local.test y root@isis.in-berlin.de
root# ctlinnd rmgroup local.test
root# _
```

Sie können INN allerdings auch damit beauftragen, diese Aufgaben für bestimmte Hierarchien automatisch zu erledigen, wenn als Absender des Artikels der dafür zuständige (d.h. gewählte) "Netzgott" auftaucht. Für die sieben großen Hierarchien ist das David Lawrence (`tale@uu.net`); für `de` ist das zur Zeit Wilhelm Bühler (`news@roka.de`). Sie können INN in der Datei `control.ct1` mitteilen, welchen Control-Messages Sie vertrauen:

```
# control.ct1 - Beispiel

# Default: mail an Administrator
all:***:mail

# newgroup
newgroup:***:mail
newgroup:news@roka.de:de.*:doit=mail
newgroup:tale@uu.net:comp.*|misc.*|news.*|rec.*|sci.*|soc.*|talk.*:doit=mail

# rmgroup
rmgroup:***:mail
rmgroup:news@roka.de:de.*:doit=mail
rmgroup:tale@uu.net:comp.*|misc.*|news.*|rec.*|sci.*|soc.*|talk.*:doit=mail
```

```
# ihave/sendme wird weggeworfen
ihave:***:drop
sendme:***:drop


# sendsys, senduuname und version sind harmlos
sendsys:***:doit=mail
senduuname:***:doit=mail
version:***:doit=mail
```

Das Beispiel zeigt eine einfache `control.ctl`-Datei. Das Flag `doit=mail` besagt, daß die betreffende Control-Message ausgeführt und anschließend eine Mail an den Administrator geschickt wird. Bei `mail` wird nur eine Mail geschickt, und bei `drop` schließlich ignoriert INN die Aufforderung ganz. An der länglichen `newgroup`-Zeile sehen Sie beispielsweise, daß der Befehl ausgeführt wird, wenn er eine Newsgruppe unterhalb der Großen Sieben betrifft und von `tale@uu.net` stammt.

## Overview-Dateien


Zum Grundrepertoire aller Newsreader gehört es, der Benutzerin eine Übersicht aller derzeit verfügbaren Artikel in einer Newsgruppe anzuzeigen. Im einfachsten Falle enthält die Liste jeweils den Absender und den Titel (d.h. das Subject) des Artikels. Manche Newsreader gruppieren außerdem noch zusammengehörige Artikel der Übersichtlichkeit wegen in sogenannten threads (Diskussions`fäden").

Das bedeutet, daß der Newsreader von jedem Artikel mindestens einmal den Header einlesen muß, um die wichtigsten Informationen zu extrahieren. Newsreader wie *tin* bearbeiten auf diese Weise alle neuen Artikel, wenn Sie sich die Übersicht über eine Newsgruppe anzeigen lassen. Das kann bei großen Gruppen schon einmal eine Minute oder länger dauern; beim Newslesen über NNTP sind auch fünf Minuten nicht unmöglich.

Um diesen etwas unbefriedigenden Zustand zu beseitigen, entwarf Geoff Collyer 1992 NOV, das News Overview System. Das ist keine eigenständige Applikation oder ähnliches, sondern ein Datenformat. Das Newssystem, in unserem Fall also INN, legt die wichtigsten Headerzeilen eines hereinkommenden Artikels in einer separaten Datei namens `.overview` im Spoolverzeichnis der jeweiligen Newsgruppe ab. Newsreader müssen jetzt nur noch eine, nämlich die Overview-Datei einlesen, um an die komplette Header-Information aller Artikel zu gelangen.  Das macht das Leben mit *tin* deutlich bequemer.

Damit INN diese Overview-Dateien erzeugt, müssen Sie in `newsfeeds` einen weiteren Feed eintragen. Er sieht üblicherweise so aus:

```
# NOV: News Overview Database
@overview:***:Tc,W0:/usr/lib/news/bin/overchan
```

Dieser Eintrag übergibt die NOV-Daten für Artikel in allen Gruppen (das Newsgruppen-Feld enthält `*`) an das Programm *overchan*. Dieses wertet die Informationen nun aus und legt sie in den entsprechenden `.overview`-Dateien ab. 

Nachdem Sie diesen Eintrag an Ihr `newsfeeds` angefügt haben, müssen Sie das System noch initialisieren. Dazu rufen Sie (als Benutzer `news`) folgenden Befehl auf:

```
news$ /usr/lib/news/expireover -a
news$ _
```

Das erzeugt für alle Artikel, die sich bereits in Ihrem Newsspool befinden, die nötigen Einträge in den `.overview`-Dateien. Anschließend veranlassen Sie *inn*d dazu, `newsfeeds` neu zu laden, so daß die



Änderung wirksam wird:

```
root# /usr/lib/news/bin/ctlinnd reload newsfeeds
root# _
```

Natürlich muß auch das Overview-System regelmäßig auf den neuesten Stand gebracht werden, d.h. in der Zwischenzeit gelöschte Artikel müssen auch aus der entsprechenden `.overview`-Datei entfernt werden. Das erledigt das Programm *expireover* für Sie:

```
news$ /usr/lib/news/bin/expireover -s
news$ _
```

## Und jetzt ohne Hände...

Nach der anfänglichen Konfiguration können Sie Ihr Newssystem fast vollständig auf Autopilot umstellen, indem Sie alle anfallenden Routine-Jobs über *cron* abwickeln lassen. Das heißt nicht, daß Sie jetzt die Hände in den Schoß legen dürfen - Sie sollten sich trotzdem von Zeit zu Zeit vergewissern, daß alles auch so läuft wie gewünscht. Dazu gehört, gelegentlich einen Blick in die Logdateien in `/var/log/news` zu werfen, ob sie irgendwelche Anomalien aufweisen, und sicherzustellen, daß genügend Plattenplatz vorhanden ist. Nichts verträgt nämlich ein Newssystem weniger als keinen Platz: dann wandern unter Umständen alle hereinkommenden Artikel kommentarlos in die Tonne. Wenn Sie Platzprobleme haben, spielen Sie etwas mit den Einstellungen in `expire.ctl` herum, kaufen sich eine neue Platte oder löschen endlich Ihre DOS-Partition: - )

Das folgende Beispiel zeigt die *crontab* für den Benutzer `news`, wie sie auf meinem System im Einsatz ist:

```
PATH=/usr/lib/news/bin:/usr/lib/news:/bin:/usr/bin
# Der Batcher wird alle 15 Minuten angeworfen:
6-52/15 * * * * send-uucp brewhq
# Falls rnews Batches rumliegen laesst:
16 * * * * rnews -U
# taegliche Wartungsarbeiten
10 8 * * * news.daily
# Artikel loeschen mit expire
28 5,15 * * * doexpire
```

Der erste Eintrag wirft alle 15 Minuten den Batcher an. Natürlich bin ich nicht so ein Vielschreiber, daß in diesem Zeitraum mehr als ein oder zwei Artikel zusammenkommen, aber mir ist es wichtiger, daß ein Artikel beim nächsten Aufbau der UUCP-Verbindung hinausgeht, als daß ich bei der Übertragung 20 Sekunden einspare.

Der Aufruf von *rnews* dient dazu, übriggebliebene Newsbatches nachträglich an *innd* einzuliefern. Das geschieht gelegentlich, wenn *innd* gedrosselt ist, während über UUCP News hereinkommen, da *rnews* die Batches dann nicht korrekt abliefern kann. Sie werden dann im Newsspool im Verzeichnis `in.coming` zwischengelagert. Das Flag `-U` weist *rnews* an, sich um solche eventuell vorhandenen Dinger zu kümmern.

Das Skript *news.daily* ist ebenfalls Teil von INN und führt tägliche Routine-Aufgaben durch; zum Beispiel stutzt es Log-Dateien zurecht und überprüft die Konsistenz des Systems. Wie der Name nahelegt, sollte es einmal täglich laufen; wenn Sie das nicht tun, wird INN das beim Systemstart feststellen und eine Mail schicken, in der es sich darüber beklagt.

Der letzte Eintrag dient dem regelmäßigen Löschen alter Artikel. Ich rufe hier nicht *expire* direkt auf, sondern ein kurzes Shell-Skript, das ich zu diesem Zweck geschrieben habe. Es sieht so aus:

```
#!/bin/sh
# /usr/lib/news/bin/doexpire - kleiner Wrapper fuer News Expiry

. /usr/lib/news/innshellvars
cd $SPPOOL

# Drosseln des Systems
ctlinnd throttle "running expire"

# Artikel wegwerfen ...
expire -s -v1

# ... und .overview-Files auf den aktuellen Stand bringen
expireover -s

# Und weiter geht's
ctlinnd go "running expire"
```

Es besetzt zunächst eine Reihe von Shell-Variablen, indem es die Datei *innshellvars* einliest. Anschließend geht es ins Spoolverzeichnis und ruft dort *expire* auf. Zu guter Letzt bringt es die in Abschnitt [7.7.10](#) beschriebenen Overview-Dateien auf den neuesten Stand.

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [Der Newsreader tin](#) **Up:** [Datenreisen und reisende Daten](#) **Previous:** [Usenet News](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

# Der Newsreader *tin*

Am Ende dieses Kapitels wollen wir noch kurz auf die Bedienung von *tin* eingehen. Er gehört wohl nicht zuletzt deswegen zu den beliebtesten Newsreadern, weil er im Vergleich zu anderen Programmen mit relativ wenigen kryptischen Tastaturkombinationen auskommt.

Wenn Sie *tin* das erste Mal aufrufen, gibt er eine ganzseitige Information aus, die kurz die Hauptfunktionen des Programms umreißt. Wenn Sie sie gelesen haben und anschließend eine beliebige Taste drücken, erscheint eine Übersicht aller Newsgruppen, die derzeit in *active* aufgeführt sind. Das sind natürlich ziemlich viele Gruppen, und es wäre sicher sehr unbequem, jedesmal zum Newslesen durch diese paar tausend Gruppen zu waten. Daher macht *tin* einen Unterschied zwischen Gruppen, die Sie immer in der Übersicht sehen wollen ("abonniert" haben), und solchen, die nur angezeigt werden, wenn Sie es wünschen. Wenn wir im folgenden also von Abonnieren und Abbestellen reden, sollten Sie sich davon nicht verwirren lassen. *tin* verändert nichts an Ihrem Newssystem - alle Artikel, die Sie von Ihrem Feed erhalten haben, sind noch da; Sie entscheiden nur, welche angezeigt werden und welche nicht.

Zu Anfang abonniert *tin* alle Gruppen, was leider etwas unhandlich ist. Wir kommen weiter unten darauf zurück, wie Sie einzelne Gruppen oder Hierarchien abbestellen können.

Suchen Sie jetzt einmal die Gruppe `comp.os.linux.answers`; Sie können dazu mit dem Cursor in dieser Gruppenübersicht hin- und herfahren. Ihnen sollte sich ein Bild wie etwa dieses bieten:

```

Group Selection (2835)                                     h=help

1327      15  comp.386bsd.announce      Announcements relating
1328      15  comp.386bsd.development   Working on 386bsd inte
1329      15  comp.os.coherent          Discussion and support
1330      15  comp.os.cpm               Discussion about the C
1331      15  comp.os.cpm.amethyst      Discussion of Amethyst
1332      15  comp.os.geos              The GEOS operating sys
1333      15  comp.os.linux             The free UNIX-clone fo
1334      42  comp.os.linux.announce     Announcements importan
-> 1335      61  comp.os.linux.answers   FAQs, How-To's, README
1336     613  comp.os.linux.development.apps Writing Linux applicat
1337     121  comp.os.linux.development.system Linux kernels, device
1338     185  comp.os.linux.x            Linux X Window System
1339    4111  comp.os.linux.setup        Linux installation and
1340     165  comp.os.linux.networking   Networking and communi
1341     141  comp.os.linux.hardware     Hardware compatibility
1342    8862  comp.os.linux.misc         Linux-specific topics
1343      11  comp.os.mach              The MACH OS from CMU

<n>=set current to n, TAB=next unread, /=search pattern, c)atchup,
g)oto, j=line down, k=line up, h)elp, m)ove, q)uit, r=toggle all/unread,
s)ubscribe, S)ub pattern, u)nsubscribe, U)nsu pattern, y)ank in/out

```

In der dritten Spalte sehen Sie die Namen der einzelnen Newsgruppen. Links davon steht die aktuelle Zahl der Artikel in dieser Gruppe, und rechts davon sehen Sie (zum Teil) die Kurzbeschreibung der Gruppe, wie sie in der Datei `newsgroups` steht. In der ganz linken Spalte numeriert *tin* die Newsgruppen durch. In den untersten drei Zeilen finden Sie einen kurzen Überblick über die wichtigsten Befehle auf dieser Ebene. Das Kommando `u` beispielsweise bestellt eine Gruppe ab, und

s abonniert eine Gruppe.

Eine vollständige Übersicht aller Befehle erhalten Sie mit dem Kommando `h`. Diese Hilfe steht Ihnen übrigens nicht nur auf der Ebene der Newsgruppen zur Verfügung, sondern auch auf allen anderen Ebenen, die wir im folgenden noch beschreiben werden.

Um die Artikel in dieser Gruppe lesen zu können, drücken Sie nun die Return-Taste. Es erscheint wieder eine Liste, die fast so ähnlich aussieht wie die erste - aber nur fast. Statt der Newsgruppen sehen Sie jetzt eine Übersicht über alle Artikel. In der Mitte finden Sie dabei das Subject der Artikel und rechts den Absender.

```
comp.os.linux.answers (54T 60A 0K 0H)                                h=help
->  1  + 1 Welcome to the comp.os.linux.* hierarchy!                Matt Welsh
    2  +   Writing and submitting a HOWTO                        Matt Welsh
    3  +   Linux Busmouse HOWTO                                  Mike Battersby
    4  + 1   Linux DOSEMU HOWTO                                  Michael E. Deisher
    5  +   Linux Distribution HOWTO (part 1/2)                  Matt Welsh
    6  +   Linux Distribution HOWTO (part 2/2)                  Matt Welsh
    7  + 1   Linux Ethernet HOWTO (part 1/2)                   Paul Gortmaker
    8  + 1   Linux Bootdisk HOWTO                               Graham Chapman
    9  +   Linux CDROM HOWTO                                    Jeff Tranter
   10  +   Linux Commercial HOWTO (part 1/2)                   Harald Milz
   11  +   Linux Commercial HOWTO (part 2/2)                   Harald Milz
   12  +   Linux Ethernet HOWTO (part 2/2)                     Paul Gortmaker
   13  +   Linux Ftape HOWTO                                    Ftape-HOWTO mainta
   14  +   Linux German HOWTO                                  Winfried Truemper
   15  +   Linux HAM HOWTO                                      Terry Dawson
   16  +   Linux HOWTO Index                                   Matt Welsh
   17  +   Linux Hardware Compatibility HOWTO                  Tawei Wan

<n>=set current to n, TAB=next unread, /=search pattern, ^K)ill/select,
a)uthor search, c)atchup, j=line down, k=line up, K=mark read, l)ist thread,
|=pipe, m)ail, o=print, q)uit, r=toggle all/unread, s)ave, t)ag, w=post
```


Wie bereits erwähnt, sortiert *tin* Artikel, die sich auf das gleiche Thema beziehen, in sogenannte Threads. Das liegt aber nicht daran, daß das Programm eine Art künstliche Intelligenz besitzt und tatsächlich den Inhalt der Artikel miteinander vergleicht, sondern es macht sich vielmehr eine bestimmte Konvention zunutze. Wenn Sie nämlich eine Antwort auf einen vorhandenen Artikel posten, übernimmt Ihr Newsreader im allgemeinen das ursprüngliche Subject und hängt den String `Re: voredran`. Das macht *tin* sich zunutze und versammelt all diese Artikel in einem gemeinsamen Thread. Der Vorteil der Sache ist natürlich, daß Sie eine Diskussion sozusagen am Stück lesen können, anstatt sich die einzelnen Beiträge mühselig zusammensuchen zu müssen. Einen Thread erkennen Sie nun daran, daß links des Subjects eine Zahl auftaucht. Sie gibt an, wieviele Folgeartikel zu dem angezeigten Artikel gehören.

Schließlich ist in dem Beispiel ganz links noch überall ein Plus zu sehen. Damit zeigt *tin* an, daß Sie den Artikel noch nicht gelesen haben.

Sie können jetzt wieder mit dem Cursor in der Artikelübersicht herumfahren. Um einen Artikel zu lesen, drücken Sie Return, wenn Sie die entsprechende Zeile erreicht haben. Über die Artikelanzeige ist nicht viel zu sagen; in den oberen drei Zeilen erscheinen noch einmal Informationen über den Absender, das Subject usw. Darunter folgt der eigentliche Text, in dem Sie mit der Leertaste vorwärts und der Taste `b` rückwärts herumblättern können.

Wenn Sie ein paar Artikel gelesen und sich an das ``Feeling" von *tin* gewöhnt haben, sollten Sie probierhalber mal einen Artikel posten. Natürlich nicht irgendwo; denken Sie daran, daß ein Testposting in einer Gruppe wie `comp.os.linux.x` mehrere Tausend Menschen erreicht. Viele Netzteilnehmer reagieren recht unfreundlich auf solche ``Beglückungen."

Für Ihren Test suchen Sie sich am besten die Gruppe `local` heraus, weil der Artikel dann gar nicht erst Ihr System verläßt. Alternativ können Sie auch eine Gruppe wie `de.test` benutzen. Machen Sie sich aber darauf gefaßt, daß sie einige Mails von sogenannten Reflektoren bekommen werden. Reflektoren sind einfache Programme, die alle Artikel, die in einer solchen Testgruppe vorbeikommen, an die Absenderin zurückschicken. So erhalten Sie Aufschluß darüber, ob, wie und auf welchen Wegen Ihre Artikel in die weite Welt gelangen.

Wenn Sie die Gruppe `local` gefunden haben, geben Sie jetzt `w` ein.  Am unteren Bildschirmrand erscheint die Aufforderung, die nach dem Subject des Postings fragt. Geben Sie `test` ein, oder was immer Sie wollen.

```
Post subject []> testing, 1 2 3_
```

Wenn Sie anschließend Return drücken, landen Sie in Ihrem Editor; meistens wird das `vi` sein. Hier können Sie den Text des Artikels eingeben.

```
Subject: testing, 1 2 3
Newsgroups: local
```

Dies ist ein Test-Artikel. Hurra.

Außer dem von Ihnen getippten Text sehen Sie am oberen Rand einen Teil des Artikel-Headers, den *tin* dem Artikel mitgeben wird. Sie sollten diese Zeilen nicht editieren, bevor Sie etwas Erfahrung mit dem Medium gesammelt haben.

Speichern Sie jetzt Ihren fertigen Text ab und verlassen den Editor. *tin* zeigt Ihnen nun, in welche Gruppen der Artikel gepostet wird. Am unteren Rand verlangt es außerdem wieder eine Eingabe von Ihnen:

```
q)uit, e)dit, p)ost: p
```

Hier können Sie sich jetzt entscheiden, ob Sie den Artikel endgültig posten wollen, ihn doch lieber noch einmal editieren, oder es ganz sein lassen. Drücken Sie also `p` und schicken Sie Ihren Artikel auf die Reise. Sie werden Ihre Festplatte jetzt kurz rattern hören, während INN den Artikel bearbeitet. Ansonsten passiert nichts.

Das hat mehrere Gründe. Zum einen schreibt INN einen Artikel nicht sofort auf die Platte, zum anderen prüft *tin* nicht andauernd, ob ein neuer Artikel angekommen ist. Warten Sie also ungefähr dreißig Sekunden und gehen anschließend in die Gruppe. Sie sollten jetzt Ihren Artikel in voller Schönheit sehen.

Als nächstes möchte ich Ihnen zeigen, wie Sie eine Antwort (ein sogenanntes Follow-up) auf einen Artikel posten können. Zeigen Sie den Artikel an, auf den Sie antworten möchten; als Kandidat bietet sich hier wieder der Testartikel von eben an. Wenn Sie jetzt `f` drücken, landen Sie wieder im Editor, um Ihre Antwort eingeben zu können. Im Unterschied zu eben zeigt der Editor jetzt bereits einen Text an, nämlich gerade den des Artikels, auf den Sie sich beziehen wollen.

```
Subject: Re: testing, 1 2 3
Newsgroups: local
References: <3jg9s6$n8t@isis.in-berlin.de>
```

Distribution:

Karla (karla@isis.in-berlin.de) wrote:

> Dies ist ein Test-Artikel. Hurra.

Dieses Zitieren (im Jargon: quoten) gehört im Usenet zum guten Ton; es hilft den Lesern, den Kontext herzustellen und herauszufinden, auf was Sie sich genau beziehen. Das heißt aber nicht, daß Sie den gesamten Text zitieren sollen; beschränken Sie sich auf das Wichtigste. Wenn Sie ein wenig News gelesen haben, werden Sie schnell herausfinden, was als schicklich gilt und was nicht. Wenn Sie das Quoten zu arg treiben, wird *tin* Sie auch darauf hinweisen und von Ihnen verlangen, daß Sie den zitierten Text etwas zusammenstreichen.

Zuletzt beschäftigen wir uns noch kurz damit, wie Sie Gruppen abonnieren oder abbestellen können. Dazu müssen Sie sich in der Gruppenübersicht befinden. Wenn Sie *u* (für *unsubscribe*) eingeben, bestellen Sie die Gruppe ab, auf der sich der Cursor gerade befindet. Das wird durch ein kleines *u* am rechten Rand angezeigt. Wenn Sie *tin* verlassen und das nächste Mal wieder aufrufen, wird die Gruppe nicht mehr angezeigt.

Wollen Sie eine Gruppe abonnieren, müssen Sie dazu den Cursor zu dieser Gruppe bewegen und *s* (*subscribe*) drücken. Aber wie kommt man zu einer Gruppe, die *tin* gar nicht anzeigt? Das geht auf zweierlei Arten. Einmal können Sie das Kommando *g* (*go*) eingeben, woraufhin *tin* Sie nach dem Namen der Gruppe fragt, zu der Sie springen wollen. Geben Sie den vollen Namen an; *tin* blendet die Gruppe daraufhin in der Liste ein, falls Sie noch nicht vorhanden war. Ist Ihnen das zu umständlich, können Sie mit dem Befehl *y* auch sämtliche vorhandenen Gruppen in die Übersicht holen. Wählen Sie jetzt die Gruppen aus, die Sie zusätzlich lesen wollen, und abonnieren Sie sie mit *s*. Wenn Sie nun wieder *y* drücken, verschwinden alle nicht abonnierten Gruppen wieder.

Die momentane benutzereigene Konfiguration von *tin* ist in der Datei `~/ .tin/tinrc` abgelegt; *tin* erzeugt sie üblicherweise beim erstmaligen Programmaufruf an und aktualisiert sie laufend. Die Bedeutung der darin einstellbaren Optionen und Standardwerte ist in der Datei selbst gut erklärt.

Geübte Anwender werden die Möglichkeit zu schätzen wissen, daß zum Abonnieren oder Abbestellen von Gruppen auch die Datei `~/ .newsrc` editiert werden kann; dabei ist jedoch Vorsicht beim Wechsel von einem zum anderen Newsreader geboten.

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------


**Next:** [Das Point-to-Point-Protokoll \(PPP\)](#) **Up:** [Datenreisen und reisende Daten](#) **Previous:** [INN](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

## Subsections

- [Voraussetzungen](#)
    - [Die andere Seite](#)
    - [Die treibende Kraft im Kern](#)
    - [Der gute Geist](#)
    - [Das Vorspiel](#)
  - [Die Konfiguration des pppd](#)
    - [Die options-Datei](#)
    - [Authentifizierung via PAP oder CHAP](#)
  - [Die Verbindung starten](#)
  - [Die Verbindung beenden](#)
- 

# Das Point-to-Point-Protokoll (PPP)

Angesichts der steigenden Verbreitung der Internetdienste und der breiteren Verfügbarkeit von Einwählknoten in das Internet stellen wir hier eine Möglichkeit vor, mit der Welt des Cyberspace in Kontakt zu treten. Dieser Abschnitt ist eine kurze Einführung in PPP  und soll eine minimale Lösung zur Anbindung ans Internet anbieten.

Binden Sie Ihren Rechner mit PPP in das Internet ein, so wird er für die Dauer der Verbindung ein Teil dieses Netzes. Das bedeutet, daß ebenso, wie Sie andere Rechner mit den entsprechenden Tools erreichen können, auch Ihr Rechner für andere erreichbar ist. Deshalb darf auf keinen Fall eine Sicherheitsüberprüfung Ihres Systems versäumt werden. Das wichtigste ist, sicherzustellen, daß alle auf Ihrem System verfügbaren Accounts durch Paßwörter geschützt sind. Ein weiteres, oft übersehenes Sicherheitsloch ist das Vermischen von Programmen, die zur Verwendung mit dem Shadow-Password-System gedacht sind, mit solchen, die ohne diese Erweiterung funktionieren. Falls Sie also auf Ihrem Rechner Shadow-Password verwenden, stellen Sie sicher, daß in `/etc/passwd` in allen Paßwortfeldern ein `*` eingetragen ist.

Wenn in einem Netzwerk wie dem Internet Hunderttausende von Rechnern verbunden sind und klaglos zusammenarbeiten sollen, so erfordert das ein hohes Maß an Disziplin von allen Systemverwaltern. Durch Ihr Linux-System und seine Fähigkeit, sich in diese heterogene Netzwerkumgebung einzufügen, obliegt Ihnen nun die Aufgabe, Ihr System verantwortungsvoll zu warten und Mißbrauch des Internets von Ihrem System aus zu verhindern. Jeder Systemverwalter ist für sein System **und die Benutzer** seines Systems verantwortlich.



# Voraussetzungen

## Die andere Seite

Zur Teilnahme am Internet gehören immer zwei. Finden Sie also Ihren Partner, der bereit ist, seine Internetverfügbarkeit an Sie weiterzugeben. Von ihm bekommen Sie:

### eine IP-Adresse

Sie ordnet den Rechner in das Netzwerk ein. Über die IP-Adresse wird der Rechner weltweit identifiziert und kann mit dieser Adresse von anderen Rechnern aus angesprochen werden. Die IP-Adresse ist eine aus vier Bytes zusammengesetzte Zahlenkombination, deren einzelne Bytes durch Punkt getrennt sind. Die IP-Adresse meines Rechners ist z. B. 193.98.158.2. Da jeder der vier Adreßteile durch ein Byte dargestellt wird, liegt jeder Teil zwischen 0 und 255. Eine Adresse 261.0.194.11 kann es deshalb nicht geben, da 261 nicht durch ein Byte dargestellt werden kann.

IP-Adressen sind ein sehr empfindlicher Teil des Internets. Verwenden Sie nur IP-Adressen, die Ihnen auch wirklich zugeteilt wurden. Es darf im Internet keine zwei verschiedenen Rechner mit derselben IP-Adresse geben.

### einen DNS-Domain-Namen

Er dient in erster Linie der Zuordnung Ihres Rechners in einen Verwaltungsbereich und ist eine besser zu merkende Bezeichnung als die IP-Adresse.

### einen Hostnamen

In Verbindung mit dem DNS-Domain-Namen ergibt sich daraus der FQDN (Fully Qualified Domain Name) Ihres Rechners. Lautet der Domain-Name z. B. „IN-Berlin.DE“ und Ihr Hostname „caesar“, so ergibt sich daraus der FQDN „caesar.IN-Berlin.DE“. DNS ist der „Domain Name Service“. Er stellt lediglich ein Übersetzungssystem dar, das FQDNs in IP-Adressen übersetzt, da sich der Name eines Rechners wesentlich einfacher merken läßt als eine IP-Adresse und gleichzeitig auch etwas über die Zugehörigkeit eines Rechners zu einem größeren Verbund aussagt.

### einen Benutzernamen

Das ist der Name, mit dem Sie sich bei Ihrem „Nachbarn“ anmelden müssen. Ein Unix-System zeigt sich grundsätzlich sehr „verschlossen“, und der einzige Unterschied zwischen einem normalen Shell-Account und einem PPP-Zugang besteht für das System, das Sie anrufen, in dem „Kommandointerpreter“, den es für Sie startet. Statt der `tcsh` oder `bash` steht in der Paßwortdatei dann eben der [pppd](#). Zu dem Benutzernamen gehört meistens auch noch ein Paßwort. Allerdings bietet das PPP selbst auch noch die Möglichkeit zu „intimen“ Fragen, so daß bei einem PPP-Zugang oft kein Paßwort nötig ist.

## Die treibende Kraft im Kern

Unter Linux ist das PPP in zwei Teile aufgeteilt. Ein Teil befindet sich im Kernel und ist ein Low-Level HDLC-Treiber, der die Bereitstellung der Daten und deren Interpretation vornimmt. HDLC steht für „High-Level Data Link Control Protocol“ und definiert das Format, in das die Datenpakete gebracht werden müssen, um zur Gegenseite verschickt werden zu können. Mit HDLC können nicht nur IP-Pakete, sondern auch alle möglichen anderen Netzwerkpakete, wie z. B. Novell- oder LAN-Manager-Pakete, verschickt werden. Dieser Treiber muß nicht besonders konfiguriert werden, er muß nur in den Kernel eingebunden sein. Dies kann entweder beim Übersetzen des Kernels geschehen,



oder er kann zur Laufzeit als Modul nachgeladen werden ([Siehe insmod](#)). Es lässt sich leicht überprüfen, ob der PPP-Treiber im Kernel enthalten ist. Das Kommando `cat /proc/net/dev` sollte ungefähr folgende Ausgabe liefern:

```
liviush:~# cat /proc/net/dev
Inter-|      Receive      |      Transmit
face  | packets  errs  drop  fifo  frame  | packets  errs  drop  fifo  colls  carrier
  lo:      0      0      0      0      0      |      0      0      0      0      0      0
  ppp0:      0      0      0      0      0      |      0      0      0      0      0      0
  ppp1:      0      0      0      0      0      |      0      0      0      0      0      0
  ppp2:      0      0      0      0      0      |      0      0      0      0      0      0
  ppp3:      0      0      0      0      0      |      0      0      0      0      0      0
liviush:~# _
```

Werden die mit ppp beginnenden Zeilen nicht ausgegeben, so ist in dem Kernel kein PPP-Treiber vorhanden. Übersetzen Sie den Kernel neu, oder laden Sie das PPP-Modul in den Kernel.

## Der gute Geist

Der zweite für die Funktion von PPP unabdingbare Teil ist ein Daemon, der `pppd`. Dieser Daemon übernimmt die Authentifizierung, die Konfiguration der Leitung und des Kernel-Treibers. Er trägt die Netzwerk-Routen in die Routing-Tabelle im Kernel ein, konfiguriert das entsprechende PPP-Device auf Ihre IP-Adresse und überwacht das Bestehen der Verbindung. Falls die Telefonverbindung zusammenbricht, löscht er automatisch die Routen, dekonfiguriert das PPP-Interface und gibt die Schnittstelle wieder frei.

## Das Vorspiel

Zwischen Ihrem „anderen Ende“ und Ihnen liegt noch etwas, das besonderer Behandlung bedarf: die Telefonleitung und die Modems. Der Aufbau der physikalischen Verbindung zwischen den beiden Rechnern wird nicht vom `pppd` übernommen. Sie kann entweder mit einem Terminalprogramm, mit `kermit` oder mit `chat` aufgebaut werden. Letzteres ist im PPP-Paket enthalten und hervorragend geeignet, um das Modem zu steuern und das Login beim anderen Rechner zu überwinden. Gleichzeitig vermeidet es den Überhang an Funktionalität, der bei einem Terminalprogramm vorhanden wäre. Ein weiterer Vorteil liegt darin, daß es vollständig automatisch einen Verbindungsaufbau bewerkstelligen kann und deshalb auch direkt vom `pppd` aus gestartet werden kann. Die Parameter, die `chat` erwartet, werden allen bekannt vorkommen, die schon einmal ein UUCP-System [konfiguriert haben](#). `chat` erwartet nämlich nur eine Folge von Zeichenketten, die den Ablauf des Verbindungsaufbaus charakterisieren, und sendet darauf eine andere Zeichenkette als Antwort. Um von zu Hause aus den Rechner im Büro anzurufen, verwende ich folgendes Chatscript:

```
chat ABORT BUSY '' ATZ OK ATDP6235097 CONNECT '' ogin: ppp ord: mypass
```

Der Aufbau des Scriptes ist denkbar einfach. Die beiden Wörter `ABORT BUSY` veranlassen `chat` dazu, den Verbindungsaufbau abubrechen, wenn es vom Modem die Antwort „`BUSY`“ bekommt. Diese Kombination kann mehrmals in der Kommandozeile verwendet werden; so könnte man auch noch `ABORT 'NO CARRIER'` und `ABORT 'NO DIALTONE'` in die Zeile einfügen, um noch weitere Fehlermeldungen des Modems abzufangen. Wichtig ist lediglich, daß Zeichenfolgen, die Leerzeichen enthalten und trotzdem als eine Zeichenkette interpretiert werden sollen, in einfache Anführungszeichen eingeschlossen werden müssen. Die leere Zeichenkette `''` veranlaßt `chat`, nicht auf eine Antwort zu warten, sondern sofort die nächste Zeichenfolge der Zeile zu senden. Auf Dauer ist es umständlich, die gesamte Kommandozeile immer von Hand einzutippen, außerdem enthält sie ja auch das Paßwort für das andere System. Am einfachsten ist es deshalb, eine Datei anzulegen, die nur

das Chatscript für chat enthält, und chat einfach mit `chat -f Dateiname` aufzurufen.

# Die Konfiguration des pppd

Um die an ihn gestellten Aufgaben erfüllen zu können, braucht der pppd eine Reihe von Informationen. Ein Teil dieser Informationen kann dem Daemon in der Kommandozeile übergeben werden, den anderen holt er sich aus seinen Konfigurationsdateien, die er im Verzeichnis `/etc/ppp` sucht. Die Dateien `options`, `pap-secrets` und `chap-secrets` sowie zwei Shellscripte namens `ip-up` und `ip-down` werden vom pppd dort gesucht. Die Datei `options` **muß** vorhanden sein, auch wenn sie leer ist.

## Die options-Datei

Die `options`-Datei ist sozusagen eine verlängerte Kommandozeile. Alle Optionen, die dem pppd in der Kommandozeile übergeben werden können, dürfen auch in der `options`-Datei verwendet werden. Wird sowieso nur eine PPP-Verbindung zu einem einzigen Rechner aufgebaut, was in den meisten Fällen wohl zutrifft, so können alle Optionen in die `options`-Datei eingetragen werden. Der Aufbau der Datei ist denkbar einfach: eine Option pro Zeile, exakt so, wie sie auch in der Kommandozeile angegeben werden kann. Beispiel:

```
/dev/cua0
38400
name livius.lunetix.de
remotename public.lunetix.de
193.98.158.8:193.98.158.12
connect chat -f /etc/ppp/chat-public
lock
modem
crtscts
defaultroute
```

Die erste Zeile legt die serielle Schnittstelle fest, über die die PPP-Verbindung laufen soll, die zweite die Geschwindigkeit. Mit der Option `name` wird der Name des eigenen Rechners angegeben, mit `remotename` der des Nachbarn. Die folgende Zeile enthält als erstes die eigene IP-Adresse und als zweiten Parameter, von dem ersten durch einen Doppelpunkt getrennt, die IP-Adresse des Nachbarn. In der sechsten Zeile wird dem pppd mitgeteilt, daß er den Verbindungsaufbau mit dem Programm `chat` bewerkstelligen soll. In der Datei `/etc/ppp/chat-public` befindet sich das Chatscript für `chat`. Um andere Programme von der seriellen Schnittstelle fernzuhalten, wird der pppd mit `lock` dazu gebracht, unter `/var/spool/uucp` ein Lockfile anzulegen. Die Optionen `modem` und `crtscts` tragen dafür Sorge, daß der pppd die DCD-Leitung des Modems überwacht, um eventuelle Verbindungsabbrüche erkennen zu können, und daß er Hardware-Handshake bei der Kommunikation mit dem Modem verwendet. Statt `crtscts` kann auch `xonxoff` oder `-crtscts` verwendet werden, um auf XON/XOFF Handshake zurückzugreifen. Die Option `defaultroute` veranlaßt den pppd, eine Standardroute für alle Netzwerkpakete zum anderen Rechner einzurichten. Diese Option ist immer dann sinnvoll, wenn der andere Rechner die einzige Verbindung zum Internet ist.

Der pppd erkennt noch eine Vielzahl anderer Optionen, die in der Manualpage `pppd(8)` beschrieben sind. Sie decken eine Fülle von Sonderfällen ab, sind aber in der Regel nicht notwendig. Die beschriebene Konfiguration sollte in 95% aller Fälle ausreichend sein, in denen ein einzelner Rechner

an das Internet angeschlossen werden soll.

## Authentifizierung via PAP oder CHAP

Zur Authentifizierung über das PAP-Protokoll ist die Datei `/etc/ppp/pap-secrets` notwendig. Die Datei hat einen vierspaltigen Aufbau. Die erste Spalte enthält den Namen des Clients, die zweite den Namen des Servers und die dritte das „Geheimnis“, auch „secret“ genannt. Die vierte Spalte ist optional und kann eine Liste von IP-Adressen enthalten, die das andere System verwenden darf. Wichtig ist hier die Verwendung der Begriffe Client und Server. Client ist immer der Rechner, von dem eine Authentifizierung verlangt wird, Server ist der, der die Authentifizierung fordert. Wichtig ist weiterhin, daß für jede PPP-Verbindung **zwei** Zeilen vorhanden sein müssen. Als Namen für Client und Server verwendet der `pppd` die mit `name` und `remotename` angegebenen Namen. Eine `pap-secrets`-Datei für obiges Beispiel sähe so aus:

```
#/etc/ppp/pap-secrets
#client          #server          #Geheimnis          #Adresse
livius.lunetix.de public.lunetix.de mein_passwort_bei_public
public.lunetix.de livius.lunetix.de publics_passwort_bei_mir
```

Die Authentifizierung via CHAP funktioniert exakt genauso, die Datei kann einfach kopiert werden. Der große Unterschied zwischen PAP und CHAP besteht darin, daß bei PAP nur eine Authentifizierung am Beginn der Verbindung stattfindet und daß die Geheimnisse unverschlüsselt über die Telefonleitung übertragen werden. Bei CHAP kann auch in regelmäßigen Abständen während der Verbindung eine Authentifizierung verlangt werden. Ferner werden bei CHAP die Geheimnisse verschlüsselt über die Leitung geschickt. Damit wird verhindert, daß Zugangsberechtigungen über die Telefonleitung abgehört werden können.

## Die Verbindung starten

Da in der beschriebenen Konfiguration schon alle Informationen, die der `pppd` benötigt, in die `options`-Datei eingetragen wurden, muß lediglich der `pppd` von der Kommandozeile aus gestartet werden.

## Die Verbindung beenden

Der `pppd` legt im Verzeichnis `/var/run` eine Datei namens `device.pid` an, in die er seine Prozeß-Id schreibt. Der Parameter `device` ist hierbei das PPP-Device, auf dem der Daemon läuft, also `ppp0`, `ppp1`, `ppp2` oder `ppp3`. Beenden läßt sich die Verbindung am einfachsten mit dem Kommando:  
`kill -INT `cat /var/run/device.pid``

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [Die Installation von Linux](#) **Up:** [Datenreisen und reisende Daten](#) **Previous:** [Der Newsreader tin](#)

# Die Installation von Linux

Linux wird auf unterschiedlichen Wegen in verschieden gestalteten Zusammenstellungen, sogenannten Distributionen, angeboten. Solche Distributionen können beliebig kopiert werden. Aus diesem Grund ist es nicht besonders schwierig, an eine aktuelle Linux-Version heranzukommen.

Wenn Sie Zugang zum Internet haben, können Sie Linux, die dafür erhältliche Freie Software und die kompletten Linux-Distributionen von vielen der öffentlichen FTP-Server beziehen. Beispielsweise finden Sie Linux auf den folgenden Rechnern in den angegebenen Verzeichnissen:

Hostname	Verzeichnis
-----	-----
ftp.uni-erlangen.de	/pub/Linux
ftp.informatik.hu-berlin.de	/pub/os/linux
ftp.informatik.rwth-aachen.de	/pub/Linux
ftp.germany.eu.net	/pub/os/Linux
nic.switch.ch	/mirror/linux

In Anbetracht des anfallenden Datentransfers von durchschnittlich mindestens 30 Megabyte für eine aktuelle Linux-Distribution ist es schon aus Kostengründen sinnvoll, die Linux-Grundausstattung auf einem anderen Wege zu besorgen. Selbst wenn Ihnen die Kosten nicht persönlich in Rechnung gestellt werden, sollten Sie auf den Bezug von Daten aus internationalen Quellen verzichten. Alle relevanten Sites werden täglich von verschiedenen Rechnern in den nationalen Netzen gespiegelt. Eine Liste der Mirrors finden Sie auf [ftp.uni-erlangen.de](#):  
/pub/Linux/LOCAL/Roadmaps/Linux-FTP-Roadmap.

Linux-Distributionen auf Diskette können Sie für sich selbst oder für Freunde kopieren. Wenn Sie die gewünschte Version nicht im Freundeskreis auftreiben können, stehen Ihnen diverse Händler mit entsprechenden Angeboten zur Verfügung. Wenn Sie sich eine Linux-Distribution kaufen, sollten Sie darauf achten, daß der Händler seine in der GNU General Public License bestimmten Pflichten erfüllt. Insbesondere müssen Sie die Quelltexte zu allen Programmen erhalten oder unter den Bedingungen der GPL angeboten bekommen.

Wegen des günstigen Preis-Leistungsverhältnisses sind Linux-Distributionen auf CD-ROM sehr beliebt. Da CD-Laufwerke in Sonderangeboten bereits für weniger als 100 DM zu bekommen sind, ist es oft attraktiver, sich ein CD-Laufwerk und die CD anstelle einer Diskettendistribution in der gleichen Preislage zu kaufen. Eine voll bespielte CD enthält so viele Daten wie 400 Disketten.

Nicht alle CDs sind als primäres Installationsmedium für Linux geeignet. Einige sind einfache Softwaresammlungen, in denen fremde Distributionen und Abzüge von FTP-Servern enthalten sind, wobei Quantität meist mehr zählt als Qualität. Es gibt aber auch homogene CD-Distributionen, die mit ausführlicher Installationsanleitung und speziellen Installationsprogrammen ausgestattet sind.

Als Alternative zu CDs und den üblichen Diskettenstapeln läßt sich Linux auch von einem

Magnetband, von einer DOS-Partition oder über ein lokales Netz installieren. Weitere Informationen hierzu finden Sie in README Dateien bei den Distributionen oder bei Ihrem lokalen Linux-Guru.

---

- [Planung](#)
  - [Eine eigene Partition für Linux finden](#)
  - [Wieviel Platz braucht Linux?](#)
  - [Linux auf mehreren Partitionen](#)
  - [Was Sie noch brauchen](#)
- [Durchführung](#)
  - [Die Installation im Überblick](#)

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [Planung](#) **Up:** [Das Linux Anwenderhandbuch](#) **Previous:** [Das Point-to-Point-Protokoll \(PPP\)](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

## Subsections

- [Eine eigene Partition für Linux finden](#)
  - [Wieviel Platz braucht Linux?](#)
  - [Linux auf mehreren Partitionen](#)
  - [Was Sie noch brauchen](#)
- 


# Planung

Wenn Sie die Linux-Distribution Ihrer Wahl neben dem Rechner liegen haben, ist es höchste Zeit, ein paar Minuten für die Planung des zukünftigen Linux-Systems zu verwenden.

## Eine eigene Partition für Linux finden

Zuerst muß ein geeigneter Platz für das neue Linux-System gefunden werden. Diese Suche gestaltet sich sehr unterschiedlich, je nachdem, ob Sie bereits ein anderes Betriebssystem installiert haben oder eine vollständig leere Festplatte vorfinden.


Obwohl es im Prinzip möglich ist, Linux von einer DOS Partition oder von CD zu betreiben, braucht Linux zur vollen Entfaltung seiner Fähigkeiten eine eigene Festplattenpartition in Ihrem Rechner.

Dabei ist Linux mit den anderen Betriebssystemen für PC sehr gut verträglich.  Die Partitionen von DOS können vollständig in das Linux-System integriert werden.

Wenn die Festplatte bereits wichtige Daten enthält, geht der sichere und empfohlene Weg zu einer Umgestaltung des Systems über eine vollständige Sicherung der Daten auf einem anderen Medium und Zurückschreiben nach erfolgter Neupartitionierung.

Wenn Sie genügend Platz im Gehäuse Ihres Rechners haben, kommt auch der Einbau einer zusätzlichen Festplatte in Frage. Falls Ihr IDE-Controller bereits mit zwei Geräten belegt ist, bietet sich der Einbau einer zweiten IDE-Karte oder eines SCSI-Hostadapters an. Letzterer kann ohne weiteres sieben und mehr Geräte betreiben, eröffnet Ihnen also wesentlich flexiblere Möglichkeiten für zukünftige Erweiterungen des Systems.

Auf einer zusätzlichen Festplatte können Sie sowohl Teile der existierenden Daten auslagern, als auch die komplette Linux-Installation unterbringen. Es ist durchaus möglich, Linux mit dem LILO Bootloader von der zweiten Festplatte zu booten. Lesen Sie zu diesem Thema auch den Abschnitt über [Bootkonzepte](#).

Wenn Sie DOS auf einer primären Partition (Laufwerk C : ) und einer erweiterten Partition (Laufwerk D : ) installiert haben, können Sie die erweiterte Partition löschen, ohne die Daten auf der primären Partition zu verlieren. Für den ungünstigeren Fall, daß die gesamte Festplatte in einer einzigen Partition unter DOS betrieben wird, besteht die Möglichkeit, nach einer Defragmentierung der Platte, mit dem Programm *fips* die Partitionsgrenze nach unten zu verschieben. 

# Wieviel Platz braucht Linux?

Der Linux-Kernel selbst braucht nur etwa 300 Kilobyte. Wenn Sie nur ein paar unentbehrliche Programme, wie `init`, `getty`, `update`, `login` und eine Shell (`bash`) installieren, kommen vielleicht 5 Megabyte für ein bootfähiges Minisystem zusammen. In bestimmten Situationen, beispielsweise wenn Sie über ein lokales Netzwerk mit den Dateisystemen anderer Rechner arbeiten können, ist das durchaus eine brauchbare Lösung.

Nach oben gibt es für die Größe eines Linux-Systems keine wirkliche Grenze. Wenn Sie mit 100 Megabyte auskommen wollen/müssen, sollten Sie sich genau überlegen, zu welchem Zweck Sie Ihr Linux-System in der nächsten Zeit einsetzen wollen und nur die dafür notwendigen Pakete installieren.

Serie	Größe (MB)
-----	-----
Basissystem	50
Entwicklungssystem	100
Emacs	25
Grafik und Sound	20
TeX	80
Netzwerk und Kommunikation	15
X Window System Basis	25
X Window System Entwicklung	35
X Window System Anwendungen	30

Die Größenangaben sind natürlich nur grobe Richtwerte. Die Summe von 365 Megabyte für eine Komplettinstallation aller Pakete einer Distribution ist aber durchaus realistisch. Hinzu kommen noch die Benutzerdaten, die bei der Arbeit mit dem System entstehen. Je nach Anwendungsgebiet kann das den Platzbedarf der Erstinstallation um ein mehrfaches übertreffen.

Die Auswahl der Serien und Pakete ist bei den verschiedenen Linux-Distributionen unterschiedlich. Alle Distributionen bieten mehr oder weniger komfortable Tools, mit denen einzelne Pakete auch nachträglich installiert werden können. Möglicherweise können bereits installierte Pakete auch wieder deinstalliert werden.

Zusätzlich zu der Partition für das Linux-Dateisystem müssen Sie auf jeden Fall eine Swappartition einplanen. Linux kann diesen Festplattenplatz als Erweiterung des Arbeitsspeichers nutzen. Wenn Ihr Rechner nicht mehr als 4 Megabyte RAM hat, ist eine Swappartition bereits zur Installation

notwendig . Die Swappartition sollte etwa doppelt so groß sein, wie das RAM. Folgen Sie zum geeigneten Zeitpunkt den Anweisungen des Installationsprogramms, um diese Partition zu aktivieren.

## Linux auf mehreren Partitionen

Wenn Sie Linux mehr Platz einräumen wollen, als Sie auf einer einzelnen Partition frei machen können, läßt sich das System auch auf mehrere Partitionen verteilen. Es ist sogar durchaus sinnvoll, bestimmte Teile des Dateisystems auf separaten Partitionen unterzubringen.

Um eine Linux-Distribution auf verschiedene Partitionen aufzuteilen, müssen Sie ein wenig über das



Konzept des Linux-Dateisystems verstehen: alle Partitionen enthalten Verzeichnisbäume, die als Äste zu einem einzigen großen Baum zusammengesetzt werden. Der Baum wird bei der Installation als Ganzes erzeugt. Um ihn auf die Partitionen zu verteilen, muß die Grundstruktur des Baumes ganz zu Anfang der Installation durch das Installationsprogramm erzeugt werden. Die einzelnen Partitionen werden dann an den von Ihnen bestimmten Verzeichnissen in diesen Baum eingehängt.

Um das System auf mehrere Partitionen zu verteilen, müssen Sie geeignete Verzeichnisse finden, an denen Sie den Baum in Äste auftrennen. Leider bietet kein Installationsprogramm eine Funktion, mit der Sie vorher ermitteln können, wie groß ein auf einem bestimmten Verzeichnis sitzender Ast des Baumes wird.

Wenn Ihnen die Installationsanleitung zu Ihrer Distribution auch keine Auskunft zu diesem Thema gibt, helfen Ihnen vielleicht die folgenden Hinweise:

- Die Hauptlast des Dateisystems liegt auf dem Verzeichnis `/usr`. Wenn Sie dieses Verzeichnis auf eine eigene Partition legen, braucht der Rest des statischen Systems (das Rootfilesystem) ca. 10 bis 15 Megabyte.
- Je nach Umfang Ihres geplanten X Window Systems liegen auf dem Verzeichnis `/usr/X11` 30 bis 90 Megabyte.
- Als eigentlichen Bereich für die Systembenutzer bietet sich eine eigene Partition für `/home` an.
- Eine TEX-Installation belegt 30 bis 80 Megabyte im Verzeichnis `/usr/lib/texmf`, je nachdem, wieviele Fonts Sie bereithalten und wo die Fontsdateien gespeichert werden.
- Wenn Sie in größerem Umfang Freie Software oder eigene Programmprojekte unter Linux übersetzen wollen, kommt eine eigene Partition für das Verzeichnis `/usr/src` in Frage.
- Wenn Sie News aus dem Usenet beziehen wollen, ist eine eigene Partition für das Verzeichnis `/var/spool/news` angebracht.

## Was Sie noch brauchen

Zur Herstellung einer Bootdiskette und zur Sicherung des Bootsektors und der Partitionstabellen brauchen Sie zwei bis vier formatierte Disketten.

Zur Sicherung eines vorhandenen Datenbestandes ist ein Bandlaufwerk am besten geeignet. Wenn Sie keins besitzen und sich auch keines leihen können, sollten Sie wenigstens die wichtigsten Dateien (Diplomarbeit, Adreßdatenbank, Korrespondenz etc.) auf Disketten oder einem anderen, von der Neuinstallation nicht betroffenen Medium speichern.

Wenn Sie besondere Hardware verwenden (Netzwerkkarte, CD-ROM, SCSI-Hostadapter), sollten Sie sich die Handbücher mit den Daten der Geräte bereitlegen. Sie brauchen eventuell die Typenbezeichnung, die Adressen des IO-Ports und die Nummer des Interrupts für diese Karten und Geräte.

Wenn Sie die grafische Benutzeroberfläche von Linux, das X Window System, benutzen möchten, brauchen Sie die Gerätebeschreibungen zu Ihrer Grafikkarte und zu Ihrem Monitor.

Wenn Sie Ihren Rechner in ein lokales Netzwerk einbinden oder ans Internet anschließen möchten, brauchen Sie einen Hostnamen, einen Domainnamen, eine IP-Adresse, eine Netzwerkadresse, die Netzmaske, die Broadcast-Adresse, die Adresse des Gateways und die Adresse des Nameservers für Ihr Netz.



Wenn Sie die Linux-Distribution per NFS von einem anderen Rechner des lokalen Netzes installieren möchten, brauchen Sie die IP-Adresse dieses Rechners und den Namen des Verzeichnisses, in dem die Distributionsdaten zu finden sind. Dieses Verzeichnis muß vom NFS-Server für den neuen Linux-Rechner exportiert werden.

Sie sollten sich ein wenig Zeit nehmen und die Installationsanleitung zu Ihrer Distribution genau durchlesen. Wenn Sie keine gedruckte Anleitung vorliegen haben, sollten Sie die verfügbaren Hilfstexte ausdrucken.

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [Durchführung](#) **Up:** [Die Installation von Linux](#) **Previous:** [Die Installation von Linux](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

## Subsections

- [Die Installation im Überblick](#)
    - [Herstellung einer Bootdiskette](#)
    - [Booten](#)
    - [Die Festplatte partitionieren](#)
    - [Swappartition und Dateisysteme einrichten](#)
    - [Das Kopieren der Daten](#)
    - [Konfiguration und Erzeugung der Bootdiskette](#)
- 

# Durchführung

## Die Installation im Überblick

Die Installation von Linux geschieht in folgenden Schritten:

1.  
Booten des Linux-Kernels.
2.  
eventuell Laden einer RAM-Disk von einer zweiten Diskette.
3.  
Partitionieren der Festplatte, Einrichten einer Swap- und einer oder mehrerer Linux-Partition(en).
4.  
Rebooten des Linux-Kernels (Wiederholung von 1. und 2.).
5.  
Initialisierung der Swappartition und Erzeugung eines Dateisystems auf mindestens einer Linux-Partition.
6.  
Kopieren und Auspacken der Daten vom Installationsmedium auf die Festplatte (die eigentliche Installation).
7.  
Minimale Konfiguration des Basissystems durch das Installationsprogramm.
8.  
Erzeugung einer Bootdiskette und/oder Installation eines bootfähigen Kernels auf der Festplatte.

Bei diesen Schritten können Sie sich von den Installationsprogrammen der Linux-Distributionen

führen lassen.

## Herstellung einer Bootdiskette

Wenn Sie sich für ein anderes Installationsmedium als eine Diskettendistribution entschieden haben, müssen Sie sich wenigstens eine Bootdiskette anfertigen, damit Sie den Rechner mit Linux starten können. Bei einigen Distributionen brauchen Sie zusätzlich eine zweite Diskette, die "Rootdiskette", auf der die Daten und Programme für die ersten Installationsschritte enthalten sind.

Diese Disketten können Sie selbst erzeugen, indem Sie die entsprechenden Daten von der CD oder einem anderen Datenträger auf leere Disketten schreiben. Die Linux-Distributionen enthalten mehr oder weniger viele Bootdisketten, die jeweils für ein bestimmtes Spektrum der denkbaren Hardwarekonfigurationen vorbereitet sind. Welche Bootdiskette für Ihr System gebraucht wird, entnehmen Sie den Hinweisen in der Installationsanleitung zur Distribution oder entsprechenden Dateien, die Sie im gleichen Verzeichnis wie die Images der Bootdisks selbst finden.

Typischerweise haben die Diskettenimages eine Größe von 1474560 Bytes, was genau der Kapazität einer rohen 3,5 Zoll Diskette entspricht. Da die 5 1/4 Zoll Disketten fast vollständig vom Markt verschwunden sind, wird dieses Format von den meisten Distributoren nicht mehr unterstützt. Wenn die Dateinamen mit den Buchstaben `.gz` enden, sind die Daten komprimiert. In diesem Fall müssen Sie sie mit dem Programm [gzip](#) entkomprimieren.


Unter Unix können Sie beispielsweise das Programm `dd` zum Übertragen der Daten von der Festplatte auf die Diskette verwenden. Wenn Sie nur DOS installiert haben, müssen Sie das Programm `kommandorawrite.exe` benutzen, um die Daten auf die Disketten zu kopieren. Ein einfacher `copy` Befehl reicht hierzu nicht aus. Nachdem Sie das Kommando `rawrite` auf dem DOS-Prompt eingegeben haben, werden Sie nach dem Namen der Datei gefragt, die geschrieben werden soll. Anschließend müssen Sie den Buchstaben für das Ziellaufwerk angeben und die Eingabe mit einem RETURN abschließen. Seien Sie vorsichtig, damit Sie nicht versehentlich auf eine Diskette mit wichtigen Daten schreiben. Eine Rekonstruktion des ursprünglichen Zustandes ist nach dem Überschreiben mit `rawrite` nicht möglich.

## Booten

Wenn Sie den Rechner mit der Linux-Bootdiskette im Laufwerk A: einschalten, erscheinen entweder die Zeichen "LILO" und anschließend ein Menü und eine Eingabeaufforderung, oder Sie sehen sofort die Meldung `Loading...`

Wenn der frisch gestartete Kernel nicht anfängt, Meldungen über die Systeminitialisierung auf den Bildschirm zu schreiben, haben Sie wahrscheinlich eine fehlerhafte Bootdiskette.

Die Meldungen werden schneller geschrieben, als Sie lesen können. Sie können aber den Bildschirm "zurückblättern", indem Sie die Tasten SHIFT und PAGEUP gemeinsam drücken.

Sollte der Kernel bei der Initialisierung hängenbleiben, können Sie anhand der Meldungen, die der Kernel auf den Bildschirm geschrieben hat, herausfinden, welche Geräte korrekt erkannt wurden und wo der Fehler aufgetreten ist. Sie finden detaillierte Informationen zum Ablauf der normalen Kernelinitialisierung ab Seite  in diesem Buch.

Wenn Ihre Installationsdiskette mit LILO bootet, können Sie dem Kernel eine Kommandozeile übergeben, mit der Sie bestimmte Parameter der Systeminitialisierung verändern können

([Bootprompt](#)).

Wenn die Systeminitialisierung erfolgreich abgeschlossen wurde, versucht Linux automatisch, eine RAM-Disk anzulegen. Bei den meisten Distributionen werden Sie dazu aufgefordert, die `rootdisk` einzulegen, damit Linux die Daten für die RAM-Disk lesen kann.

Nachdem auch dieser Schritt erfolgreich abgeschlossen wurde, erscheint zum ersten Mal der Login-Prompt. Wie Sie sich bei dem nun laufenden Linux-System anmelden können und wie Sie das Installationsprogramm aufrufen, erfahren Sie aus der Begrüßungsmeldung oder aus der Installationsanleitung.

## Die Festplatte partitionieren

Nachdem Sie bei der Vorbereitung für die Neuinstallation Ihres Linux-Systems einen geeigneten Festplattenbereich ausgewählt haben, müssen Sie diesen Bereich zu Beginn der Installation partitionieren. Sie benötigen mindestens eine Partition für das Linux-Dateisystem und eine zweite Partition als Swap-Bereich.

Die Installationsprozeduren der Linux-Distributionen unterstützen Sie bei dieser Aufgabe. Wenn Sie mit dem Partitionierungsprogramm eines anderen Betriebssystems vertraut sind, können Sie die Partitionierung in der Regel auch mit diesem Programm durchführen. Sie müssen in diesem Fall lediglich die Partitionskennung unter Linux ändern, damit das andere Betriebssystem, unter dem Sie die Partitionen eingerichtet haben, nicht versehentlich versucht, auf das Linux-Dateisystem zuzugreifen.

In diesem Buch finden Sie eine ausführliche Beschreibung des Systemprogramms [fdisk](#), mit dem Sie die Partitionierung unter Linux durchführen können.

Jede Veränderung an einer Festplatte, die schon Daten enthält (z. B. eine MS-DOS Installation), kann zum Verlust der Daten führen! Deshalb sollte vor jeder Veränderung an den Partitionstabellen ein Backup aller Daten gemacht werden.

Wenn die Partitionierung abgeschlossen ist, muß der Rechner neu gebootet werden.

## Swappartition und Dateisysteme einrichten

Wenn die Festplatte partitioniert und der Rechner neu gebootet ist, kann das Installationsprogramm mit der Einrichtung der Dateisysteme und der Swappartition fortfahren. Bei allen Linux-Distributionen auf dem Markt kann für die Linux-Partitionen nur der Dateisystemtyp EXT2 gewählt werden.

Sie müssen in einem Menü die Festplattenpartitionen selektieren, die Sie für die Installation von Linux vorgesehen haben. Alle vorher auf diesen Partitionen gespeicherten Daten gehen bei der Installation des neuen Systems verloren.

Wenn Sie das Linux-Dateisystem über mehrere Partitionen verteilen wollen, müssen Sie zu jeder zusätzlichen Partition das Verzeichnis angeben, an das diese Partition angehängt werden soll.

Sie haben die Möglichkeit, bei der Erzeugung eines neuen Dateisystems die Festplattenpartition auf defekte Blöcke untersuchen zu lassen. Dieser Vorgang ist zeitaufwendig und bei neuen Festplatten eigentlich unnötig. Wenn auf Ihrer Festplatte defekte Blöcke gefunden werden, sollten Sie auf dieses Gerät besser ganz verzichten.

# Das Kopieren der Daten

Spätestens dann, wenn das Dateisystem gemacht ist, setzt die interaktive/automatische Installationsprozedur der Distribution ein. Die Installationsprogramme erklären sich selbst.

Sie können verschiedene Serien und einzelne Pakete aus den Serien zur Installation auswählen oder ablehnen. Je nachdem, welche Distribution Sie installieren, erhalten Sie kurze Informationstexte über den Inhalt der Pakete auf deutsch oder auf englisch angezeigt. Bestimmte Pakete sind für die Funktion des gesamten Systems notwendig, deshalb werden sie automatisch installiert.

Achten Sie darauf, daß Sie nicht mehr Pakete zur Installation auswählen, als Ihre Festplatte aufnehmen kann. Wenn Sie mehrere Partitionen benutzen, müssen Sie die einzelnen Dateisysteme so zusammensetzen, daß jede Partition ihren Ast des Verzeichnisbaums aufnehmen kann. Wenn Sie nicht sicher sind, wieviel Platz die ausgewählten Daten belegen, sollten Sie am Anfang lieber auf ein paar Pakete verzichten. Sie können jedes Paket auch zu einem beliebigen späteren Zeitpunkt installieren.

## Konfiguration und Erzeugung der Bootdiskette

Nachdem alle von Ihnen ausgewählten Daten auf die Festplatte kopiert sind, werden bereits durch das Installationsprogramm die wichtigsten Konfigurationen am neuen System vorgenommen. In der Datei `/etc/fstab` werden die Partitionen eingetragen, auf denen das neue System installiert wurde.

Abschließend wird eine neue Bootdiskette erstellt, mit der Sie das neue System booten können. Alternativ können Sie sich auch durch das Installationsprogramm den LILO Bootloader installieren lassen, mit dem Sie Linux direkt von der Festplatte starten können.

Wenn Sie das neue Linux-System zum ersten Mal gestartet haben, können Sie sich nur als `root` mit den Privilegien der Systemverwalterin einloggen. Es ist wichtig, daß Sie sich gleich zu Anfang Ihrer Arbeit mit dem neuen System einen eigenen Account einrichten, der mit normalen Benutzerrechten ausgestattet ist. Von da an sollten Sie immer unter diesem Account arbeiten und den `root`-Account nur dann benutzen, wenn Sie zur Ausführung eines Kommandos tatsächlich die Privilegien brauchen.

Wenn Sie diesen Rat nicht annehmen, verzichten Sie auf eine wertvolle Fähigkeit Ihres neuen Betriebssystems: nur, wenn Sie mit normalen Benutzerrechten arbeiten, kann Linux die unabsichtliche Veränderung oder Löschung von Daten verhindern.

Wenn Sie das C-Entwicklungssystem und die Quelltexte zum Linux-Kernel installiert haben, sollten Sie sich einen neuen Kernel generieren, der genau Ihrer Systemkonfiguration entspricht.

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [Dateisysteme](#) **Up:** [Die Installation von Linux](#) **Previous:** [Planung](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

# Dateisysteme

Für die Leistungsfähigkeit eines Betriebssystems ist die Verwaltung der Massenspeicher von großer Bedeutung. Massenspeicher, das sind vor allem Festplatten und Disketten, neuerdings aber auch zunehmend CD-ROMs.

Auf unterster Ebene sind diese Medien in Speicherblöcke einheitlicher Größe unterteilt. Um auf Dateien unterschiedlicher Größe einzeln zugreifen zu können, ist eine weitere Strukturierung des Mediums notwendig. Die Organisation der Dateien in Verzeichnissen ist auf der Benutzerebene so verbreitet, daß sie selbstverständlich erscheint. Trotzdem gibt es bei der internen Realisierung dieser Struktur enorme Unterschiede.

Bei Linux treten diese Unterschiede in einer Vielfalt auf, die man fast schon als Wildwuchs bezeichnen möchte. Es gibt mindestens vier eigene reguläre Linux-Dateisysteme; dazu kommen mehrere von anderen Architekturen übernommene Dateisysteme und ein Pseudodateisystem ohne dauerhaftes Medium (das Prozeßdateisystem).

Vor der Entscheidung für oder gegen die Installation eines der angebotenen Dateisysteme auf der Festplatte oder einer Diskette muß sich die Systemverwalterin zuerst einen Überblick verschaffen. Es ist Ziel dieses Kapitels, die dazu notwendigen Begriffe zu bilden und die Zusammenhänge zu veranschaulichen. Dazu werden zuerst anhand des von MINIX übernommenen Dateisystems die grundsätzlichen Funktionen beschrieben. Danach werden die wichtigsten Komponenten der regulären Linux-Dateisysteme dargestellt. Zum Abschluß werden noch kurz die zusätzlichen Dateisystem-Varianten für Linux beschrieben. Diese Beschreibungen sollen Ihnen einen Eindruck vermitteln, welche Möglichkeiten das Dateisystem-Konzept von Linux bietet. Eine ausführliche Erklärung aller Details finden Sie nicht hier, sondern in den verschiedenen Dokumenten zu den jeweiligen Source-Paketen.

- 
- [Das Minix-Dateisystem](#)
    - [I-Nodes](#)
    - [Verzeichnisse](#)
    - [Superblock und Bitmaps](#)
    - [Links](#)
  - [Die neuen Dateisysteme](#)
    - [Die Zeitmarken in der I-Node](#)
    - [Entwicklung](#)
    - [Das zweite erweiterte Dateisystem \(ext2fs\)](#)

- [Das xiafs](#)
- [Bewertung](#)
- [Spezialdateisysteme](#)
  - [Das Prozeßdateisystem](#)
  - [Das ISO-9660-Dateisystem](#)
  - [umsdos](#)

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [Das Minix-Dateisystem](#) **Up:** [Das Linux Anwenderhandbuch](#) **Previous:** [Durchführung](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

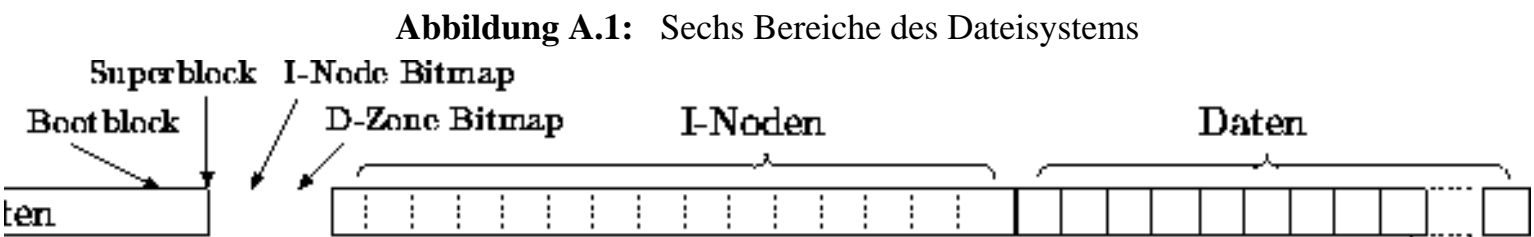
Subsections

- [I-Nodes](#)
- [Verzeichnisse](#)
- [Superblock und Bitmaps](#)
  - [Datenzonen und Plattenblöcke](#)
- [Links](#)
  - [Links auf Verzeichnisse](#)

# Das Minix-Dateisystem

Sein erstes Dateisystem hat Linux von Minix ``geborgt''. Dieses Minix-Dateisystem wird ``traditionell'' auch heute noch als Standard für Linux eingesetzt, es wird wenigstens theoretisch von allen Linux-Installationen unterstützt. Seine einfache Struktur macht es zum bevorzugten Objekt dieser Erklärung.

Der Massenspeicher, sei es eine Diskette oder eine Festplattenpartition, stellt sich dem Betriebssystem zuerst als eine lineare Kette gleichgroßer Blöcke dar. Typischerweise sind diese Blöcke genau ein Kilobyte, das sind 1024 Bytes, groß.



Das Minix-Dateisystem unterteilt diese Kette in sechs Bereiche. Der Aufbau ist in [Abbildung A.1](#) dargestellt.

Die Proportionen der Bereiche sind für ein Minix-Dateisystem auf einer 3,5-Zoll-HD-Diskette dargestellt. Der Bereich für die Daten ist dabei stark verkürzt. Bei einer Gesamtzahl von 1440 Datenblöcken werden jeweils ein Block als Bootblock, Superblock, als I-Node-Bitmap und als Datenzonen-Bitmap verwendet. Weitere 15 Blöcke enthalten die I-Nodes, der gesamte Rest steht für Daten zur Verfügung.

Abbildung



### A.3:

Minix

Superblock

### Abbildung

### A.2:

Minix-I-Node

# I-Nodes

Der logistische Kern des Dateisystems sind die I-Nodes (Informations-Knoten). Für jede Datei existiert eine dieser Datenstrukturen, deren zentrale Komponenten die Zeiger auf die eigentlichen Datenzonen der Datei sind. Die Datenzonen bestehen aus einer konstanten Anzahl Datenblöcke. Ebenso wie die Blöcke sind auch die Zonen wie eine Kette aneinandergereiht. Damit sind die Datenzonen über ihre Nummer adressierbar. Ein Zeiger auf eine Datenzone enthält also die Nummer der Datenzone als Adresse. Wenn ein Zeiger unbenutzt ist (zum Beispiel, weil die Datei 0 Bytes, also keine Daten enthält), ist die Adresse Null.

### Abbildung

### A.4:

Direkte

und

Indirekte

Zuordnung

der

Datenzonen


zu

einer

Datei

Die Datenzonen können theoretisch mehrere Blöcke groß sein, bei der normalen Konfiguration entsprechen sie genau einem Block. Damit wird für jedes angefangene Kilobyte Daten ein Zeiger benötigt.

Um die Verwaltung der Datenstruktur einfach zu halten und damit die Performance des Dateisystems zu verbessern, sind alle I-Nodes gleich groß (beim Minix-Dateisystem sind es 32 Bytes). Mit der festgelegten Größe ist zwangsläufig auch die Anzahl der Zeiger in der I-Node beschränkt. Beim Minix-Dateisystem lassen sich höchstens sieben Datenzonen direkt aus der I-Node adressieren. Weil die Beschränkung auf eine Dateigröße von 7 Kilobyte inakzeptabel ist, werden weitere Datenzonen indirekt adressiert. Dazu gibt es einen Zeiger auf einen Datenblock, der nichts anderes als Zeiger auf die eigentlichen Datenzonen enthält.

Bei der Realisierung von Zonenzeigern durch vorzeichenlose Zweibytezahlen (unsigned short) können mit einem indirekten Zeigerblock 512 Datenzonen adressiert werden , was die maximale Dateigröße auf

immerhin 519 Kilobyte erhöht. Weil diese Einschränkung aber immer noch unbefriedigend ist, gibt es die doppelt indirekte Adressierung. Der doppelt indirekte Zeiger zeigt auf einen Datenblock, der nur Zeiger auf einfach indirekte Zeigerblöcke in der oben besprochenen Art enthält. Auf diese Weise lassen sich 262663 Datenzonen adressieren. Das sind über 256 Megabyte, die aber wegen der Einschränkung des Minix-Dateisystems auf 64 Megabyte Partitionen nicht ausgenutzt werden können.

Die Abbildung [A.4](#) stellt diese Verbindung zwischen der I-Node und den Datenzonen dar.

Außer den Zeigern auf die Datenzonen enthalten die I-Nodes noch andere Informationen. Hier sind der Eigentümer und die Gruppe der Datei mit den jeweiligen Zugriffsrechten (Permissions) gespeichert, es gibt Informationen über den Dateityp, die Größe, die Anzahl der Links und das Datum der letzten Veränderung (mtime). Der Kernel liest bei jedem Zugriff eines Benutzers auf eine Datei die entsprechende I-Node und stellt fest, ob dieser Benutzer die erforderlichen Rechte für den angeforderten Zugriff hat. Nur, wenn der Zugriff erlaubt ist, wird die entsprechende Lese- oder Schreiboperation durchgeführt.

## Verzeichnisse


Die auf der Oberfläche das Dateisystem darstellenden Verzeichnisse sind Dateien eines speziellen Typs. Jedes Verzeichnis hat eine I-Node, in der Eigentümer und Zugriffsrechte für das Verzeichnis genau wie für eine normale Datei gespeichert sind. Das Verzeichnis selbst, das heißt die Einträge, werden in (mindestens) einem Datenblock gespeichert, der ebenso wie bei einer normalen Datei durch einen Zonenzeiger adressiert wird. Wenn ein Verzeichnis größer als 7 Kilobyte wird, werden weitere Datenblöcke indirekt adressiert; auch darin unterscheidet sich ein Verzeichnis nicht von einer normalen Datei.

Jeder Eintrag in ein normales Minix-Verzeichnis wird in einem 16 Byte großen Feld gespeichert. Die ersten zwei Bytes enthalten die Nummer der zur Datei gehörenden I-Node, sie sind also ein Zeiger auf eine I-Node. Die darauffolgenden vierzehn Bytes enthalten den Dateinamen. Wenn der Dateiname kürzer als vierzehn Zeichen ist, werden die restlichen Bytes durch Nullen aufgefüllt.

### Abbildung

#### A.5:

Verzeichnis  
im  
Minix  
FS

Eine Linux-spezifische Erweiterung des Minix-Dateisystems ermöglicht die Verwendung von Dateinamen bis zu einer Länge von 30 Zeichen, wenn das `mkfs.minix`-Kommando bei der Erzeugung des Dateisystems mit der entsprechenden Option (`-n30`) aufgerufen wurde. 

Jedes Verzeichnis enthält mindestens zwei Einträge, den Punkt und den Doppelpunkt. Der Zeiger des Doppelpunkt-Verzeichnisses zeigt auf die I-Node des übergeordneten Verzeichnisses, der Eintrag mit dem einfachen Punkt zeigt auf die I-Node des Verzeichnisses selbst. Diese beiden Einträge sind nötig, um ein Verzeichnis oder eine Datei relativ vom aktuellen Verzeichnis aus benennen zu können, zum Beispiel

um ein übergeordnetes Verzeichnis anzuzeigen. 

# Superblock und Bitmaps

Der erste Block einer Diskette oder einer Partition kann ein Programm zum Laden des Betriebssystems, den sogenannten Boot-Loader, enthalten. Deshalb heißt dieser Block Bootblock. Er wird beim Einschalten des Rechners gelesen und ausgewertet, das eigentliche Dateisystem fängt erst mit dem zweiten Block, dem Superblock, an.


Der in Abbildung [A.3](#) dargestellte Superblock des Minix-Dateisystems enthält Daten über den Typ und den Aufbau des Dateisystems. Mit einer magischen Zahl unterscheidet das Betriebssystem die unterschiedlichen Dateisystemtypen.


Durch die Anzahl der I-Nodes und die Anzahl der Datenzonen ist die Größe des Dateisystems bestimmt.

Weitere Strukturen, die zur Verwaltung des Minix-Dateisystems verwendet werden und deren Größe im Superblock festgehalten wird, sind die sogenannten Bitmaps. Es gibt je eine Bitmap für die I-Nodes und für die Datenzonen. Jeder I-Node beziehungsweise jeder Datenzone ist ein Bit in der entsprechenden Bitmap zugeordnet. Wenn die Zone oder die I-Node benutzt ist, wird das entsprechende Bit gesetzt.

Wenn der Kernel eine freie I-Node oder eine freie Datenzone braucht, weil eine neue Datei angelegt werden soll oder eine bestehende erweitert wird, kann mit diesen Bitmaps sehr schnell die nächste freie Stelle gefunden werden.

## Datenzonen und Plattenblöcke

Die Datenzonen des von Linux verwendeten Minix-Dateisystems entsprechen in ihrer Größe genau den Plattenblöcken. Deshalb ist die begriffliche Unterscheidung vielleicht etwas verwirrend, jedenfalls erscheint sie unnötig. Die interne Realisierung dieser Trennung erlaubt es aber, mehr als einen Plattenblock je Datenzone zu verwenden,  indem der Superblock entsprechend verändert wird.

Auf diese Weise könnte die Beschränkung des Minix-Dateisystems auf 64 Megabyte je Partition überwunden werden. Weil aber die Veränderung der Zonengröße auf zwei oder mehr Blöcke eine Verschwendung von kostbarem Plattenplatz wäre,  ist mit den neuen Linux-Dateisystemen ein anderer Weg eingeschlagen worden.

Die Anzahl der Blöcke je Datenzone ist im Superblock nicht direkt gespeichert, weil das Betriebssystem diese Zahl niemals verwendet. Stattdessen ist der Logarithmus dieser Zahl im Superblock festgehalten. Diese Zahl gibt die Anzahl der Schiebeoperationen (shifts) an, die notwendig sind, um aus einer Zonenadresse die entsprechende Blockadresse zu ermitteln.

## Links

Wie bereits oben erklärt wurde, bestehen die Einträge eines Verzeichnisses aus dem Dateinamen und einem Zeiger auf die zu der Datei gehörende I-Node. Der einzige Zugang zu einer Datei wird durch die

I-Node hergestellt. Dieser festgelegte Weg erlaubt es auf einfache Weise, mehrere Verzeichniseinträge für ein und dieselbe Datei zu erzeugen.

Die Verbindung einer I-Node (und der damit über die Zonenzeiger verbundenen Datei) mit einem Verzeichniseintrag wird als Link bezeichnet. Mit dem `ln`-Kommando (`→` Seite [□](#)) kann ein solcher Link auf eine bestehende Datei erzeugt werden. Das Kommando erzeugt einfach einen neuen Eintrag im aktuellen oder einem anderen angegebenen Verzeichnis und benutzt den I-Node-Zeiger der bestehenden Datei.

Weil die I-Node Bestandteil des Dateisystems einer in sich geschlossenen Partition ist, können solche „harten“ Links nur innerhalb einer einzigen Partition bzw. einer einzigen Diskette erzeugt werden. Um Links über die Partitions Grenzen hinweg zu ermöglichen, bietet Linux sogenannte symbolische Links an. Die symbolischen Links werden als spezieller Dateityp im Verzeichnis eingetragen und belegen im Fall des Minix-Dateisystems vier Datenzonen. In diesen Zonen ist der Zugriffsweg auf die Datei gespeichert. Es ist auch möglich, daß ein symbolischer Link ins Leere zeigt, zum Beispiel, wenn die Datei, auf die er zeigen sollte, gelöscht wurde.

## Links auf Verzeichnisse

Wie die Beispiele des Punkt- und Doppelpunkt-Eintrags in jedem Verzeichnis zeigen, sind harte Links auf Verzeichnisse theoretisch möglich. Es ist aber nicht erlaubt, zusätzliche harte Links auf ein Verzeichnis anzulegen.

Zusätzliche, harte Links auf ein Verzeichnis würden die Baumstruktur der Verzeichnishierarchie zerstören und könnten leicht zu endlosen Rekursionen führen.

Die bessere, erlaubte Lösung zum Erzeugen zusätzlicher Zugänge für Verzeichnisse sind symbolische Links.

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [Die neuen Dateisysteme](#) **Up:** [Dateisysteme](#) **Previous:** [Dateisysteme](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)




## Subsections

- [Die Zeitmarken in der I-Node](#)
  - [Entwicklung](#)
  - [Das zweite erweiterte Dateisystem \(ext2fs\)](#)
    - [Das Valid-Flag](#)
    - [Dateiattribute](#)
    - [Gruppierung der Datenzonen](#)
  - [Das xiafs](#)
- 

# Die neuen Dateisysteme


Das Minix-Dateisystem weist einige wichtige Eigenschaften eines Unix-Dateisystems auf. Die Vereinfachungen, die es einerseits zu einem hervorragenden Lehrbeispiel machen, erweisen sich auf der anderen Seite aber als ein Mangel. Vor allem die Beschränkung auf 64 Megabyte je Partition ist für das umfangreiche Linux-System äußerst unpraktisch. Die Beschränkung der Dateinamen auf 14 Zeichen ist ebenfalls unbefriedigend. Die Unterstützung nur einer Zeitmarke (mtime) entspricht darüberhinaus nicht dem POSIX-Standard, der drei Zeitmarken für eine Datei vorsieht.

## Die Zeitmarken in der I-Node

Eine Zeitmarke, in der automatisch das Datum und die Uhrzeit der letzten Veränderung einer Datei gespeichert wird, gibt es bei den meisten Betriebssystemen (mtime). Diese Zeitmarke ermöglicht beispielsweise die Unterscheidung verschieden aktueller Versionen der gleichen Datei. Die von Unix bekannte und in den neuen Linux Dateisystemen unterstützte Speicherung von zwei zusätzlichen Zeitmarken, das Datum der letzten Statusänderung  und das Datum des letzten Zugriffs, gibt den Dateien zusätzliche interessante Merkmale. Mit geeigneten Kommandos wie `find` ( $\rightarrow$  Seite ) und `ls` ( $\rightarrow$  Seite ) kann so beispielsweise festgestellt werden, ob ein Brief bereits gelesen wurde oder ob ein Text bereits überarbeitet ist.

Die Vorteile mehrerer Zeitmarken können übrigens auch bei den einfachen Dateisystemen, die nur eine Zeitmarke speichern, beobachtet werden, weil der Kernel intern im sogenannten virtuellen Dateisystem (virtual file system, vfs) eine abstrakte Verallgemeinerung aller I-Nodes benutzt. Solange sich die I-Node im Speicher befindet, werden alle Zeitmarken festgehalten und benutzt. Erst wenn die I-Node auf die Platte geschrieben werden muß, weil der Arbeitsspeicher für eine neue Aufgabe benötigt wird, gehen die zusätzlichen Zeitmarken verloren.

# Entwicklung

Der Weg zu einem neuen Dateisystem für Linux ist mit dem "extended filesystem" (extfs) von Remy Card begonnen worden. Dieses Dateisystem erlaubt bereits Partitionsgrößen von 2 Gigabyte und Dateinamen mit einer Länge von bis zu 255 Zeichen. Freie I-Nodes und Datenzonen werden im extfs durch verkettete Listen anstelle der Bitmaps verwaltet. 

Die Entwicklung eines "passenden" Dateisystems für Linux war mit dem extfs nicht zu Ende. Der experimentelle Charakter des extfs ist von Anfang an betont worden und seine Ablösung lange angekündigt.

Das neue, zweite erweiterte Dateisystem (ext2fs) hat sich als das Linux-Dateisystem der Zukunft durchgesetzt. In seiner 128 Bytes großen I-Node enthält es neben den erwarteten Ergänzungen, wie zum Beispiel der Unterstützung aller von POSIX definierten Zeitmarken, eine Reihe ganz neuer Attribute, die erst zum Teil implementiert sind und zu einem größeren Teil Raum für zukünftige Entwicklungen lassen.

Neben diesem sehr weitsichtig angelegten Modell gibt es noch weitere Entwicklungen, von denen das von Q. Frank Xia entwickelte xiafs eine gewisse Bedeutung erlangt hat.


## Das zweite erweiterte Dateisystem (ext2fs)

Die Haupteigenschaften des ext2fs sind:

- Maximale Partitionsgröße ist 2 Gigabyte. Die Erweiterung auf bis zu 4 Terabyte ist in der Struktur bereits vorgesehen.
- Die zulässige Länge für Dateinamen beträgt 1-255 Zeichen.
- Die maximale Größe einer Datei beträgt 16 Gigabyte.
- Es werden alle drei von POSIX festgelegten Zeitmarken für I-Nodes unterstützt (atime, ctime, mtime).
- Eine zusätzliche Zeitmarke für die Löschezit ermöglicht zukünftigen Spezialprogrammen das Retten versehentlich gelöschter Dateien.
- Ein Eintrag in der I-Node kann zukünftig vom NFS-Server zur Unterscheidung verschiedener Versionen einer Datei verwendet werden.
- Zwei zusätzliche Einträge in der I-Node sind für zukünftige Erweiterungen der Zugriffskontrolle vorgesehen. (Access Control List)
- Die Unterstützung fragmentierter Datenzonen mit mehr als einem Block ist vorgesehen.
- Ein Teil des Dateisystems kann dem Superuser (root) zur Benutzung vorbehalten werden.
- Es können 12 Datenzonen direkt adressiert werden.
- Die Unterteilung des Dateisystems in Gruppen erlaubt die Rettung von Daten auch bei Schäden in sensiblen Bereichen.
- Die symbolischen Links bis zu einer Länge von 60 Zeichen belegen keine Datenzonen mehr, sondern speichern die Information allein in der I-Node (fast symbolic link).
- Ein spezielles Flag im Superblock erleichtert das Auffinden fehlerhafter Dateisysteme.

- Das ext2fs ist derzeit das einzige Linux-Dateisystem, bei dem die automatische Synchronisierung von Cache und Festplatte bei jeder Veränderung der sensiblen Daten von Superblock und I-Nodes implementiert ist. Zusätzlich können normale Dateien auch zum vollständig synchronen Schreiben geöffnet werden.
- Ein spezielles Dateiattribut veranlaßt den Kernel, eine Datei beim Löschen mit zufälligen Daten zu überschreiben, um so die Rekonstruktion der Datei sicher zu verhindern.

Die Abbildung [A.6](#) stellt die I-Node des ext2fs dar.

Viele der Komponenten haben ihre Entsprechung in der Minix-I-Node. Der auffällige Größenunterschied resultiert nur zum Teil aus den vielen zusätzlichen Einträgen. Vor allem die Vergrößerung der Zonenzeiger von zwei auf vier Bytes trägt einen weiteren Teil bei. Diese Verdoppelung ermöglicht auf der einen Seite die Adressierung von bis zu 4 Terabyte großen Partitionen.  Auf der anderen Seite können in einem indirekten Block nur noch 256 Zeiger untergebracht werden, was die Einführung eines dreifach indirekten Blockes sinnvoll gemacht hat. Der dreifach indirekte Block enthält Zeiger auf bis zu 256 weitere doppelt indirekte Blöcke, die jeweils wieder Zeiger auf einfach indirekte Blöcke enthalten.

Der Eintrag für die Gruppenkennzahl ist bereits seit dem ersten erweiterten Dateisystem auf zwei Bytes (unsigned short) gewachsen.

**Abbildung**  
**A.6:**  
 Extended  
 2  
 FS  
 I-node

## Das Valid-Flag


Ein spezielles Byte im Superblock, das Valid-Flag, wird vom Kernel beim Aufsetzen einer ext2fs-Partition gelöscht. Wenn das Dateisystem beim regulären Systemhalt wieder abgesetzt wird, setzt der Kernel das Valid-Flag wieder. Bei einem Systemabsturz bleibt das Flag selbstverständlich ungesetzt. Indem es vor dem Aufsetzen eines ext2fs dieses Flag prüft, kann das Betriebssystem feststellen, ob es sich um ein gültiges (valid) Dateisystem handelt, und nötigenfalls eine Warnung ausgeben.

Das `e2fsck`-Programm testet ebenfalls dieses Flag, bevor es das Dateisystem überprüft, und überspringt die Prüfung gültiger Dateisysteme, wenn es nicht durch die entsprechende Option dazu gezwungen wird. Auf diese Weise kann das `e2fsck`-Kommando automatisch beim Booten des Systems mit den Bootutilities aufgerufen werden, bevor das Dateisystem zusammengesetzt wird, ohne daß deshalb nach jedem regulären Systemhalt gleich das ganze Dateisystem durchgekämmt werden muß.



Seit der Kernel-Version 0.99.10 wird auch das Root-Filesystem beim regulären Systemhalt mit den anderen Dateisystemen zusammen abgesetzt und das Valid-Flag gesetzt.

Zusätzlich zu der passiven Indikatorfunktion kann das Valid-Flag auch vom Kernel benutzt werden, um ausdrücklich ein fehlerhaftes Dateisystem zu markieren. Beim Aufsetzen eines ext2fs führt der Kernel immer eine minimale Konsistenzprüfung durch. Wenn bei diesem Test ein Fehler festgestellt wird, setzt der Kernel das Valid-Flag auf einen speziellen Wert, um damit dem `e2fsck` und den Bootutilities die Notwendigkeit eines Dateisystem-Checks zu signalisieren.

Außer dem Valid-Flag wird im Superblock des ext2fs noch die maximale Anzahl normaler Mounts festgehalten . Beim Aufsetzen eines ext2fs wird automatisch ein Zähler im Superblock erhöht; wenn dieser Zähler größer wird als die festgelegte Maximalzahl, wird beim automatischen Aufruf durch die Bootutilities ein File-System-Check durchgeführt, sonst wird eine Warnung ausgegeben.

## Dateiattribute

In den I-Nodes des ext2fs steht ein vier Bytes großes Feld für ``Flags" zur Verfügung. Von diesem 32 Bit großen Bereich ist zur Zeit nur die Bedeutung der ersten sieben Bits für bestimmte Dateiattribute definiert, der Rest des Feldes steht für zukünftige Erweiterungen bereit.

Die sieben definierten Attribute haben folgende Bedeutung:

a

(append) Eine mit diesem Attribut gekennzeichnete Datei kann nur durch Anhängen zusätzlicher Daten verändert werden. Das überschreiben der bereits gespeicherter Daten, das Löschen, Umbenennen oder Verschieben ist nicht möglich.

d

(dump) Mit diesem Attribut können Dateien markiert werden, die von der inkrementellen Sicherung durch `dump` ausgenommen werden sollen.

i

(immutable) Eine Datei, die mit diesem Attribut ausgestattet ist, läßt sich in keiner Weise verändern. Sie kann nicht gelöscht oder umbenannt werden, es können keine Links auf sie erzeugt werden und der Inhalt der Datei läßt sich nicht überschreiben oder erweitern.

Zum Setzen oder Löschen dieses Attributs sind Rootprivilegien erforderlich.

s

(secure) Durch dieses Attribut können Dateien, deren Inhalt besonders vor dem unberechtigten Zugriff anderer Systembenutzer geschützt werden soll, zum sicheren Löschen markiert werden.

Die Datenblöcke werden dann beim Löschen durch zufällige Daten überschrieben. 

S

(Sync) Dieses Attribut veranlaßt den Kernel, jede Veränderung der I-Node synchron durchzuführen. Die veränderten Daten werden ungepuffert sofort auf das Speichermedium geschrieben.

u

(undelete) Noch ohne Funktion. Durch dieses Attribut soll eine Datei zukünftig auch nach dem Löschen noch intakt gehalten werden, damit das Retten der Daten durch die noch zu entwickelnde `undelete`-Funktion möglich ist.



(compressed) Noch ohne Funktion. Dieses Attribut wird einmal den Kernel veranlassen, die entsprechende Datei komprimiert zu speichern.

Die Dateiattribute können Sie mit dem Kommando `lsattr` für alle Dateien eines ext2-Verzeichnisses anzeigen lassen. Zum Ändern der Attribute gibt es das `chattr`-Kommando. Veränderungen an den Dateiattributen können nur die Benutzer vornehmen, die auch die Datei selbst verändern dürfen (also schreibberechtigt sind).

## Gruppierung der Datenzonen

Das ext2fs benutzt nicht die beim Minix-Dateisystem vorgestellte einfache Unterteilung der Partition in sechs Bereiche, sondern es unterteilt eine Partition zusätzlich in sogenannte Gruppen. In der aktuellen Version des ext2fs ist jede der Gruppen genau 8192 Blöcke groß; eine variable Gruppengröße ist für zukünftige Versionen vorgesehen. Die Gruppen enthalten jeweils eine Kopie des Superblockes sowie einen dem Superblock in seiner Funktion ähnlichen Gruppenblock, die Bitmaps und die I-Nodes für die Datenblöcke. Damit werden die einzelnen Gruppen fast so verwaltet wie ganze Partitionen im Minix-Dateisystem.

Neben dem Geschwindigkeitsvorteil, den die räumliche Nähe der I-Nodes zu den Datenblöcken vor allem bei großen Partitionen bringt, erhöht diese Unterteilung die Datensicherheit. Weil auf der Partition mehrere Kopien des Superblocks an genau definierten Stellen zu finden sind, kann der eventuell beschädigte erste Superblock vom `e2fsck`-Programm oder von dem speziell zu diesem Zweck entworfenen `mksuper`-Kommando repariert werden. Außerdem kann dem `mount`-Kommando als zusätzliche Option beim Mounten eines Ext2-Dateisystems eine Blocknummer angegeben werden, aus der es dann den Superblock liest.

## Das xiafs

Das von Frank Q. Xia entwickelte xiafs ist eine Weiterentwicklung des Minix-Dateisystems. Die hervorstechenden Eigenschaften sind:

- Reservierter "Systembereich" für den Kernel und Loader im Bootsektor erlauben das Booten von Festplatte.
- Partitionen bis zu zwei Gigabyte werden unterstützt, Erweiterung möglich.
- Dateinamen bis zu einer Länge von 255 Zeichen sind zugelassen.
- Alle drei von POSIX in der `stat`-Struktur vorgesehenen Zeitmarkierungen werden unterstützt.
- Variable Zonengröße, acht direkte Zonen, je ein indirekter und doppelt indirekter Block (Dateigröße bis 64 Megabyte bei 1 Block/Zone).
- Zonen- und I-Node-Verwaltung mit Bitmaps.

Besonders interessant ist die Bootfähigkeit von primären xiafs Partitionen der ersten Festplatte. Der Kernel wird in einem reservierten Bereich am Anfang der Partition gespeichert und automatisch geladen, wenn die Partition aktiviert ist. Der Kernel und der Loader werden mit dem `mkboot` Programm installiert. Zusätzlich kann das `mkboot` Programm einen Master Boot Loader installieren,

der die Auswahl zwischen mehreren Bootpartitionen erlaubt. 

---

[Next](#) [Up](#) [Previous](#) [Contents](#) [Index](#)

**Next:** [Bewertung](#) **Up:** [Dateisysteme](#) **Previous:** [Das Minix-Dateisystem](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

# Bewertung

Wenn auf einer Linux-Distribution mehr als ein Dateisystem zur Installation angeboten wird, steht die Systemverwalterin vor der Qual der Wahl. Ist ein Dateisystem erst einmal installiert und in Benutzung, kann es nicht ohne weiteres umgebaut werden. Hier soll und kann Ihnen nicht die eigene Entscheidung für eines der Dateisysteme abgenommen werden. Jedes Modell hat seine spezifischen Vor- und Nachteile. Eine Zusammenfassung der wichtigsten Aspekte kann Ihnen die Entscheidung aber sicher erleichtern.

Eine absolute Entscheidung ist nicht unbedingt notwendig. Die Dateisysteme sind miteinander gut verträglich, es können also mehrere Partitionen mit unterschiedlichen Dateisystemen zu einem Dateisystembaum zusammengesetzt werden. Allerdings vergrößert die Unterstützung jedes einzelnen Dateisystems den Kernel, damit wird der verbleibende Arbeitsspeicher für die Anwendungen kleiner.

Außer für Disketten, die auch weiterhin häufig mit dem Minix-Dateisystem ausgerüstet werden, weil es wenig Verwaltungs-Overhead hat und uneingeschränkt durch alle Linux-Versionen unterstützt wird, fällt die Wahl wohl auf eines der neuen Dateisysteme.

Ein schwerwiegendes Kriterium ist die Funktionssicherheit eines Dateisystems. Das ext2fs hat mit der SLS-Distribution sehr weite Verbreitung gefunden und konnte sich so in der Praxis als äußerst zuverlässig beweisen. Deshalb, und weil Remy Card weiterhin am Ausbau der geplanten Features seines Dateisystems arbeitet, hat es sich zum De-Facto-Standard für Linux entwickelt.

Die vielen neuen Features und Erweiterungsmöglichkeiten geben ihm ein zukunftsorientiertes und luxuriöses Image. Allerdings bringen die Veränderungen immer auch die Gefahr neuer Bugs mit sich. Außerdem verbraucht das ext2fs mit seinen großen I-Nodes den meisten Platz für Verwaltungsstrukturen.

Das xiafs ist im Vergleich zum ext2fs einige Wochen älter und es enthält weniger Erweiterungen. Es hat sich in der Praxis aber nicht richtig durchsetzen können und ist deswegen nicht so gut ausgetestet. Trotzdem kann ihm eine gute Stabilität bescheinigt werden. Der in das Dateisystemkonzept integrierte Bootloader ist ein wichtiger Wertungspunkt für das xiafs.

Subsections

- [Das Prozeßdateisystem](#)
  - [Die Prozeßverzeichnisse](#)
- [Das ISO-9660-Dateisystem](#)
- [umsdos](#)

# Spezialdateisysteme

## Das Prozeßdateisystem

Das Prozeßdateisystem von Linux stellt zur Laufzeit des Betriebssystems Daten aus dem Kernel in der Form eines normalen Dateisystems dar. Es belegt dabei keinen Platz auf der Festplatte, die Verzeichnisse und Dateien existieren allein im Arbeitsspeicher des Kernels.

Wie jedes andere Dateisystem kann/muß auch das Prozeßdateisystem mit dem Systemprogramm mount auf ein existierendes Verzeichnis im Dateisystembaum aufgesetzt werden. Auch wenn im Prinzip jedes beliebige Verzeichnis als Aufsetzpunkt erlaubt ist, wird in der Regel das Verzeichnis /proc verwendet.

```

# mount -t proc proc /proc
# ls -F /proc
1/          31328/      95/         dma          modules
10215/      42/           96/         filesystems  net/
11809/      4383/         9787/        interrupts   self/
18226/      4384/         9788/        kcore        stat
29911/      44/           9791/        kmsg         uptime
29916/      46/           9792/        ksyms        version
30/         48/           9793/        loadavg
31/         52/           devices      meminfo
# _
    
```

Normalerweise wird das Prozeßdateisystem zusammen mit den anderen Dateisystemen in der Datei /etc/fstab eingetragen und beim Systemstart automatisch gemountet. Sie können aber selbstverständlich auch das Prozeßdateisystem manuell aufsetzen, wie im Beispiel oben gezeigt. In jedem Fall finden Sie dann im Verzeichnis /proc eine Menge Unterverzeichnisse und ein paar Dateien.

Mit Hilfe des Prozeßdateisystems stellt Linux eine universelle Schnittstelle zu allen relevanten Kerneldaten zur Verfügung. Auf diesem Weg können alle Programme auf diese Daten zugreifen, ohne direkt im Kernelspeicher zu lesen. Das hat einerseits Vorteile im Hinblick auf die Datensicherheit, andererseits erspart es auch die sonst notwendige Verwaltung einer Symboltabelle, in der die Speicheradressen der gesuchten Kerneldaten verzeichnet sind.

Neben den Daten über den Status des laufenden Systems besitzt der Kernel eine Menge Information über die Hardware des Rechners. In einigen Dateien des Prozeßdateisystems finden Sie deshalb eine komplette Hardwareanalyse.

### **cmdline**

Diese Datei enthält die Kommandozeile, mit der der Kernel von LILO gestartet wurde. Aus dieser Datei können alle Bootparameter gelesen werden, die auf dem Bootprompt eingegeben wurden.

### **cpuinfo**

In dieser Datei sind alle Informationen über den Typ und die Leistung der CPU enthalten.

### **devices**

Diese Datei enthält die Hauptgerätenummern und die Namen der aktiven Gerätetreiber im Kernel. Wenn Sie nicht sicher sind, welche Geräte von dem gerade laufenden Kernel unterstützt werden, finden Sie in dieser Datei die eine vollständige Liste. Das Programm MAKEDEV wertet die Information in dieser Datei aus, um beim Update die Gerätedateien für alle unterstützten Devices zu erzeugen.

### **dma**

gibt eine Liste der belegten DMA-Kanäle mit den sie belegenden Geräten an.

### **filesystems**

In dieser Datei steht eine Liste aller vom Kernel unterstützten Dateisystemtypen. Zukünftige Versionen von make werden diese Information nutzen, um ein unbestimmtes Dateisystem selbständig zu mounten.

### **interrupts**

gibt eine Liste aller belegten Hardwareinterrupts aus. Nach der Interruptnummer wird die Anzahl der ausgelösten Unterbrechungen und die Bezeichnung des auslösenden Gerätes ausgegeben. Ein + vor dem Gerätenamen zeigt an, daß es sich um einen ``schnellen" Interrupt handelt.

### **ioports**

In dieser Datei finden Sie eine Liste aller belegten Adressen des IO-Bereichs. Hinter den Adressbereichen, die in Hexadezimalzahlen angegeben sind, steht jeweils die Bezeichnung des Gerätetreibers, der sich für diesen Bereich registriert hat.

### **kcore**

ist der Zugang zum gesamten Arbeitsspeicher des Rechners. Diese Datei kann nur mit Root-Privilegien gelesen werden.

### **kmsg**

enthält die Kernmeldungen, wenn sie nicht durch den Protokollschreiber syslogd in eine andere Datei umgelenkt werden.

### **ksyms**

zeigt die exportierten Kernelsymbole für die Modulschnittstelle.

### **loadavg**

gibt drei Zahlen aus, die anzeigen, wieviele Prozesse durchschnittlich innerhalb der letzten 1, 5 und 15 Minuten gelaufen sind. Diese Zahlen werden normalerweise vom Programm uptime ausgegeben.

### **locks**

Diese Datei enthält die Liste aller aktiven File-Locks.

## **meminfo**

wird normalerweise vom `free`-Kommando ausgewertet und zeigt die Auslastung des Arbeitsspeichers und des Swap-Space an.

## **modules**

enthält eine Liste der zur Laufzeit in den Kernelspeicher geladenen und noch darin befindlichen Kernelmodule. In jeder Zeile der Datei stehen der Modulname, die Anzahl der vom Modul belegten Speicherseiten (4 Kilobyte je Seite), gegebenenfalls die Namen der Module, die mit Symbolen dieses Moduls gelinkt sind und die Anzahl der Prozesse, die das Modul benutzen. Der Inhalt dieser Datei wird normalerweise mit dem `lsmod`-Kommando ausgegeben.

## **mounts**

Diese Datei enthält eine Liste aller aktuell gemounteten Dateisysteme. Der Inhalt ist identisch mit `/etc/mtab`.

## **pci**

Bei Rechnern mit PCI-Bus enthält diese Datei Hardwareinformationen zu allen Geräten, die an diesem Bus betrieben werden.

## **rtc**

Wenn die Kernelunterstützung für die "Real Time Clock" (CMOS-Uhr) aktiviert wurde, gibt diese Datei alle Daten dieses Gerätes aus.

## **scsi**

Dieses Verzeichnis ein Unterverzeichnis für den SCSI-Hostadapter und eine Datei mit dem Namen `scsi`, in der alle erkannten Geräte aufgelistet sind.

## **stat**

enthält die aktuelle Statusinformation des Kernels.

## **sys**

Die Unterverzeichnisse dieses Verzeichnisses enthalten verschiedene Informationen über das Laufzeitsystem.

## **uptime**

zeigt zwei Zahlen, die erste sagt Ihnen, wie viele Sekunden das System läuft, die zweite Zahl zeigt die Anzahl der Sekunden, die der Rechner im Idle-Prozeß gelaufen ist.

## **version**

enthält den Linux-Banner mit der Versionsnummer und dem Übersetzungsdatum des laufenden Kernels.

# **Die Prozeßverzeichnisse**

Die meisten Unterverzeichnisse sind mit bloßen Nummern benannt. Diese Nummern entsprechen den Prozeß-IDs der aktuell in der Prozeßtabelle eingetragenen Prozesse, in den Verzeichnissen finden Sie alle relevanten Daten zu dem jeweiligen Prozeß.

Das Verzeichnis `self` ist ein Link auf das Prozeßverzeichnis des aktuellen Prozesses. Hier kann also jedes Programm auf seine eigenen Daten zugreifen, ohne seine eigene Prozeßnummer zu kennen.

```
$ ls -F /proc/self/  
cmdline  environ  fd/      mem      stat  
cwd@     exe@     maps    root@    statm  
$ _
```

### **cmdline**

enthält die komplette Kommandozeile für den Prozeß, wenn der Prozeß nicht vollständig in den Swapbereich ausgelagert ist.

### **cwd**

ist ein Link auf das aktuelle Arbeitsverzeichnis des Prozesses.

### **environ**

enthält die Daten aus der Prozeßumgebung.

### **exe**

ist ein Link auf die ausführbare Programmdatei des Prozesses.

### **fd**

enthält Links auf alle offenen Dateien des Prozesses.

### **maps**

zeigt den Adreßbereich, die Attribute und gegebenenfalls den Offset, die Gerätenummern und I-Node der mit mmap eingeblendeten virtuellen Speicherbereiche.

### **mem**

ist ein Zugang zum Speicherbereich des Prozesses.

### **root**

ist ein Link auf das Root-Verzeichnis des Prozesses.

### **stat und statm**

enthalten Statusinformationen zum Prozeß aus der Prozeßtabelle. Diese Daten werden normalerweise mit dem ps-Kommando angezeigt.

## **Das ISO-9660-Dateisystem**

Im ISO-9660-Standard ist das systemunabhängige Dateisystem für CD-ROMs beschrieben.

Wegen ihrer großen Kapazität (ca. 600 Megabyte) sind CD-ROMs als Medium für die Verbreitung von Daten sehr interessant. Unter anderem wird auch ein großer Teil der Freien Software auf CD-ROM angeboten, nicht zuletzt auch Linux und die Linux-Distributionen.

Damit es von allen Betriebssystemen verstanden wird, ist die Spezifikation der allgemeinsten Stufe des ISO-9660 Dateisystems so etwas wie der kleinste gemeinsame Nenner aller relevanten Betriebssysteme. Wie bei DOS dürfen die Dateinamen maximal 8 Zeichen lang sein. Eine Erweiterung von drei Zeichen ist für normale Dateien erlaubt, es dürfen nur Großbuchstaben verwendet werden.

Um die weit über die Beschränkungen von DOS hinausgehenden Fähigkeiten der UNIX-Dateisysteme

auch mit CDs nutzen zu können, gibt es eine mit dem Standard konforme Erweiterung von ISO-9660, die sogenannte Rock-Ridge-Erweiterung, mit denen außer den längeren Dateinamen auch Eigentümer, Zugriffsrechte, Links und anderes mehr im CD-Dateisystem verwaltet werden können.

Linux kann sowohl mit CDs der allgemeinen ersten Stufe des ISO-Standards als auch mit den Rock-Ridge-Erweiterungen umgehen, vorausgesetzt, Sie verwenden einen der vielen von Linux unterstützten CD-Spieler.


Übrigens sind auch die Foto-CDs im ISO-Standard formatiert, Sie können also auch die Bilddaten unter Linux verwenden.

## umsdos

Das umsdos-Dateisystem ermöglicht die Benutzung von normalen DOS-Dateisystemen mit den speziellen Features eines Unix-Dateisystems. Insbesondere können in einem umsdos-System die Dateien Eigentümer und Gruppen mit allen dazugehörenden Rechten haben, es können Links angelegt werden, und es sind Dateinamen mit einer Länge bis zu 255 Zeichen erlaubt.

Diese Erweiterung zum DOS-Dateisystem ermöglicht es, Linux allein von einer DOS-Partition zu starten.

Weil die für die funktionale Erweiterung notwendigen Daten nicht in der Struktur des DOS-Dateisystems vorgesehen sind, müssen sie von umsdos im normalen Datenbereich verwaltet werden. Dazu dient die spezielle Datei `-linux- .--`, die für jedes umsdos-Verzeichnis existieren muß. Nur wenn es diese Datei gibt, werden die entsprechenden Erweiterungen für dieses Verzeichnis wirksam. Ohne die spezielle Datei werden alle Dateizugriffe unverändert an das DOS-Dateisystem weitergereicht.

Um die Datei `-linux- .--` zu erzeugen, müssen Sie unter Linux das `umssync`-Kommando benutzen. Jede Veränderung des Verzeichnisses wird dann automatisch vom umsdos-Treiber in der Datei vermerkt. Wenn Sie unter Linux ein Unterverzeichnis in einem umsdos-Verzeichnis anlegen, wird dieses Unterverzeichnis automatisch auch von umsdos verwaltet. Wenn Sie unter DOS Veränderungen an Verzeichnissen vornehmen, die unter Linux mit umsdos verwaltet werden, müssen Sie beim nächsten Start von Linux wieder das `umssync`-Kommando aufrufen, damit die Datei `-linux- .--` aktualisiert wird. Sollten Sie Teile Ihrer Festplatte regelmäßig sowohl unter DOS als auch unter Linux mit umsdos benutzen, ist es sinnvoll, das `umssync`-Kommando beim Systemstart automatisch aus einer `rc`-Datei (→ Seite ) auszuführen.

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [Die Linux-Console](#) **Up:** [Dateisysteme](#) **Previous:** [Bewertung](#)



# Die Linux-Console

Das schönste Betriebssystem ist wertlos und die vielen pfiffigen Programme ohne Nutzen, wenn Sie als Benutzer keine Daten in den Computer eingeben und die Ergebnisse der Datenverarbeitung nicht wahrnehmen können. Deshalb werden die beiden wichtigsten Geräte der Mensch-Maschine-Schnittstelle, die Tastatur und der Bildschirm, fast schon mit dem Computer an sich identifiziert. Die räumliche Trennung von Rechner und Bildschirmarbeitsplatz, wie sie beispielsweise bei seriellen Terminals üblich ist, verdeutlicht das wirkliche Verhältnis: Tastatur und Bildschirm sind selbständige Geräte, die vom Betriebssystem verwaltet werden.

- 
- [Der Bildschirm](#)
    - [Einen neuen Zeichensatz laden](#)
  - [Die Tastatur](#)
    - [Die Tastaturtabelle](#)
    - [Metazeichen](#)

## Subsections

- [Einen neuen Zeichensatz laden](#)
    - [Reset der Console](#)
- 

# Der Bildschirm

Der Monitor selbst wird nicht vom Betriebssystem angesteuert, das ist Sache der Grafikkarte. Um ein Zeichen auf dem Bildschirm darzustellen, gibt Linux bestimmte Befehle an die Grafikkarte, die dann entsprechende Signale an den Monitor sendet.

Beim einfachen Textbildschirm können nur Zeichen aus einem beschränkten Zeichensatz (Font) dargestellt werden. Jedes dieser Zeichen ist aus einzelnen Bildpunkten aufgebaut. Wie so ein Zeichen im Einzelnen aussieht, wird zunächst nicht vom Betriebssystem, sondern von der Grafikkarte bestimmt. Der Zeichensatz ist dort in einem nicht flüchtigen Speicher (ROM) gespeichert.

Nach dem Einschalten arbeitet Linux also im Prinzip mit dem gleichen Zeichensatz wie MS-DOS. Dieser Zeichensatz hat zwar mit dem IBM-PC eine sehr weite Verbreitung, in einer internationalen, offenen Systemwelt hat er aber keinen Bestand. Für den Austausch von Daten zwischen Unix-Systemen im Internet sind die Zeichensätze nach der ISO-Norm besser geeignet. Aus diesem Grund ist Linux standardmäßig nach dem ISO-Latin1-Zeichensatz ausgerichtet, in dem zu den 126 internationalen ASCII Zeichen noch die Sonderzeichen der mittel-, nord- und südeuropäischen Länder enthalten sind.

Um diese Zeichen darzustellen, benutzt der Kernel eine Übersetzungstabelle, die die ISO-Latin1-Zeichen in die entsprechenden Codes des PC-Zeichensatzes umsetzt. Weil der PC-Zeichensatz nicht alle Buchstaben von ISO-Latin1 enthält, hat dieser Standardzeichensatz von Linux viele Lücken, die durch das PC-Zeichen Nummer 254 dargestellt werden.

### Abbildung:

Die  
Zeichensätze  
für  
die  
Linux-Console  
(1.  
Teil)

Linux stellt noch drei weitere Übersetzungstabellen zur Verfügung. Eine dient zur Darstellung von vt100-Grafikzeichen, eine andere stellt den PC-Zeichensatz identisch dar und die letzte kann frei definiert werden, indem vom Benutzer mit dem `mapscrn(1)`-Kommando eine Tabelle aus dem Verzeichnis `/usr/lib/kbd/consoletrans` geladen wird.

**Abbildung:** Die Zeichensätze für die Linux-Console  
(1. Teil)

4 c	CONTROL-O	CONTROL-N			
1	ISO-Latin1	→	PC	ESC ( B	ESC ) B
2	vt100-Grafik	→	PC	ESC ( 0	ESC ) 0
3	PC	→	PC	ESC ( U	ESC ) U
4	benutzerdefiniert	→	PC	ESC ( K	ESC ) K

Die aktuelle Tabelle wird durch einen von zwei Zeigern bestimmt, zwischen denen für jedes virtuelle Terminal separat mit `CONTROL-O` und `CONTROL-N` umgeschaltet werden kann. Jeder der beiden Zeiger kann mit den in der Tabelle gezeigten `ESC`-Sequenzen auf eine der vier Tabellen gesetzt werden.

**Abbildung:**  
Die  
Zeichensätze  
für  
die  
Linux-Console  
(2.  
Teil)

Per Default ist der zweite Zeiger auf die vt100-Grafiktabelle eingestellt. Deshalb erscheint ein typisches Durcheinander von Strichgrafiken und Großbuchstaben, wenn versehentlich eine Binärdatei mit `CONTROL-N` auf den Bildschirm geschrieben wurde. Sie erhalten das alte Schriftbild wieder, indem Sie mit `CONTROL -O` auf den ersten Zeiger zurückschalten.

Wenn Sie mit einer der `ESC`-Sequenzen einen der Zeiger auf eine andere Tabelle setzen, gilt diese Übersetzung für alle virtuellen Terminals, die aktuell mit diesem Zeiger arbeiten.

## Einen neuen Zeichensatz laden

Wenn Ihnen der PC-Zeichensatz nicht genügt, können Sie mit dem `setfont`-Kommando einen

komplett neuen Zeichensatz in den Kernel laden. Dabei werden die Pixelbilder der einzelnen Zeichen ersetzt. Sie können damit den vollständigen ISO-Latin1-Zeichensatz oder auch hebräische oder kyrillische Schriftzeichen laden.

Die Fontdateien befinden sich im Verzeichnis `/usr/lib/kbd/consolefonts`. Um beispielsweise den vollständigen ISO-Latin1-Zeichensatz zu aktivieren, sind die folgenden Kommandos nötig:

```
$ setfont iso01.f16
Loading 8x16 font from file /usr/lib/kbd/consolefonts/iso01.f16
$ mapscrn trivial
Loading symbolic screen map from file /usr/lib/kbd/consoletrans/trivial
$ echo "^[K"
```

```
$ _
```

Nach dieser Einstellung haben Sie auf der Console genau den gleichen Zeichensatz wie in einem `xterm`.

## Reset der Console

Jedes virtuelle Terminal kann durch ein `reset` in einen definierten Zustand (zurück-) gebracht werden. Wenn kein Kommando dieses Namens vorhanden ist, kann dasselbe Ergebnis durch `setterm -reset` oder durch Eingabe von `ESC-c` erreicht werden.

---

Next	Up	Previous	Contents	Index
------	----	----------	----------	-------

**Next:** [Die Tastatur](#) **Up:** [Die Linux-Console](#) **Previous:** [Die Linux-Console](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

## Subsections

- [Die Tastaturtabelle](#)
    - [Die Belegung der keycodes](#)
    - [Die Definition der Funktionstasten](#)
    - [Die zusammengesetzten Zeichen \(Diacriticals\)](#)
  - [Metazeichen](#)
- 

# Die Tastatur

Wenn Sie auf Ihrer Tastatur (Keyboard) die Taste mit der Aufschrift `Z' drücken, erwarten Sie normalerweise, daß auf dem Bildschirm das Zeichen `Z' erscheint. Das gleiche gilt natürlich auch für die Benutzerin in den USA, in Finnland oder in Moskau. In vielen Ländern werden üblicherweise Tastaturen benutzt, die auf die sprachlichen Eigenarten der jeweiligen Schriftsprache ausgerichtet sind. Bekanntermaßen enthält das deutsche Tastaturlayout die Umlaute ä , Ä , ö , Ö , ü , Ü und ß. Außerdem sind die Tasten im Vergleich zu dem in den USA üblichen Keyboard anders angeordnet: Z und Y sind vertauscht, : ; / etc. befinden sich an anderen Stellen.

## Die Tastaturtabelle

Damit Linux mit allen möglichen Tastaturen zusammenarbeiten kann, wird auch im Tastaturtreiber eine Tabelle zur Umsetzung der rohen Tastatureingabe in den länderspezifischen Zeichencode eingesetzt.

Für die deutsche Tastatur mit Umlauten ist die Tabelle `de-latin1.map` vorgesehen, die im Verzeichnis `/usr/lib/kbd/keytables` zu finden ist. Die Tabelle wird mit dem `loadkeys`-Kommando in den Kernel geladen:

```
$ loadkeys de-latin1
Loading /usr/lib/kbd/keytables/de-latin1.map
$ _
```

Die Tastaturtabelle besteht aus drei Abschnitten.

## Die Belegung der keycodes

Im ersten Abschnitt werden bestimmten ``keycodes" symbolische Namen für die Zeichen zugeordnet, die beim Drücken der den keycode erzeugenden Taste eingegeben werden sollen.

Der keycode wird vom Kernel aus dem von der Tastatur kommenden scancode ermittelt. Beide Codes können Sie mit dem `showkey`-Kommando für jede Taste ermitteln.

Mit symbolischen Namen können sowohl die üblichen Schriftzeichen als auch die Funktions- und Sondertasten benannt werden. Eine vollständige Liste der verwendbaren symbolischen Namen erhalten Sie mit dem Kommando `dumpkey -long-info`.

Es ist üblich, die Tasten mehrfach zu belegen. So sind alle Buchstabentasten gleichzeitig mit dem großen und dem kleinen Schriftzeichen belegt; die Umschaltung erfolgt mit der `SHIFT`-Taste.

Zusätzlich zu `SHIFT` werden noch `ALTGR`, `CONTROL` und die linke `ALT`-Taste zum Umschalten auf zusätzliche Tastaturbelegungen verwendet. Durch Kombinationen dieser vier Tasten lassen sich alle Tasten mit 16 symbolischen Namen belegen.

Die Reihenfolge der Zuordnung ergibt sich, wenn Sie den Umschalttasten in der Reihenfolge `SHIFT`, `ALTGR`, `CONTROL` und `ALT` die Werte 1, 2, 4, 8 zuordnen. Das erste Symbol, das einem keycode zugeordnet wird, erscheint, wenn keine der Umschalttasten gedrückt ist, das zweite erscheint zusammen mit `SHIFT`, das dritte mit `ALTGR`, das vierte mit `SHIFT ALTGR` usw. ...

Es müssen nicht alle 16 erlaubten Belegungen für eine Taste aufgeführt werden. Ohne Belegung geben die Tastenkombinationen normalerweise keine Zeichen aus. Zur vereinfachten Eingabe der "normalen" Schriftzeichen gibt es jedoch einen speziellen Mechanismus von `loadkeys`, mit dem eine Standardreihe von Belegungen erzeugt wird: wenn ein keycode nur mit einem einzigen ASCII-Buchstaben belegt wird, werden die korrespondierenden Zeichen für Tastenkombinationen mit `SHIFT`, `CONTROL`, `ALT` und `ALT CONTROL` automatisch erzeugt.

Es ist möglich, bestimmte, einzelne Tastenkombinationen zu belegen, indem der symbolische Name durch die Bezeichnungen der Umschalttasten eingeleitet wird, mit denen das dem symbolischen Namen entsprechende Zeichen erzeugt werden soll.

## Die Definition der Funktionstasten

Einige der symbolischen Namen für Tastaturbelegungen bezeichnen keine Schriftzeichen, sondern Funktionen, die normalerweise den Funktionstasten zugeordnet werden. Diese Tasten erzeugen normalerweise Sequenzen von mehreren Zeichen, die von Anwenderprogrammen abgefangen und besonders ausgewertet werden können.

Im zweiten Abschnitt der Keymap werden die Zeichensequenzen für die Funktionstasten als "string" den symbolischen Namen zugeordnet.

## Die zusammengesetzten Zeichen (Diacriticals)

Durch die symbolischen Namen `dead_grave`, `dead_acute`, `dead_circumflex`, `dead_tilde` und `dead_diaeresis` kann beliebigen Tastenkombinationen die Funktion "toter" Vorzeichen gegeben werden. Wenn so ein Zeichen eingegeben wird, erscheint es nicht sofort in der Ausgabe, sondern wird zusammen mit dem darauffolgenden Tastendruck interpretiert. Wenn das so entstandene Zeichenpaar mit einem der -im dritten Abschnitt der Keymap hinter dem Schlüsselwort `compose` aufgeführten -Paare

übereinstimmt, wird es entsprechend dieser compose-Anweisung ersetzt. Stimmt das Zeichenpaar mit keinem der compose-Paare überein, wird es unverändert weiterverarbeitet.

# Metazeichen

Die oben erwähnte automatische Ergänzung der Tastenbelegung für normale Buchstaben führt zur Erzeugung von ``Metazeichen'', wenn diese Tasten zusammen mit der linken ALT-Taste gedrückt werden. Derselbe Effekt kann für alle Tasten durch direkte Zuordnung der symbolischen Namen ``Meta_*` erzielt werden.

Die Metazeichen werden vom Kernel auf eine von zwei Arten weiterbearbeitet. Erstens kann das Zeichen automatisch als `ESC`- Sequenz ausgegeben werden, bei der der dem Metazeichen zugrundeliegende Buchstabe durch das `ESCAPE`-Zeichen eingeleitet wird.

Zweitens kann das Zeichen aus dem ``unteren'' Bereich der Codetabelle (0-127) in den oberen Bereich transponiert werden, indem das höchstwertige Bit gesetzt wird. Auf diese Weise können die Sonderzeichen aus diesem Teil der Tabelle eingegeben werden.

Zwischen den beiden Arten der Nachbearbeitung der Metazeichen kann durch das Kommando `setmetamode ( 1 )` mit den Argumenten `esc` oder `meta` umgeschaltet werden.

Die automatische Erzeugung von `ESC`-Sequenzen ist vorteilhaft für alle Programme, die mit solchen Steuerkommandos arbeiten. Beispielsweise erlauben die `bash` oder der `emacs` so die Benutzung der linken ALT-Taste als Metataste.

## Abbildung

### B.3:

Die  
deutsche  
MF2-Tastatur

---

[Next](#) [Up](#) [Previous](#) [Contents](#) [Index](#)

**Next:** [Locales und Native Language](#) **Up:** [Die Linux-Console](#) **Previous:** [Der Bildschirm](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

# Locales und Native Language Support

Wenn Linus Torvalds den Funktionen seines Betriebssystems finnische Namen gegeben hätte und alle Kommentare und Meldungen in Finnisch erscheinen würden, hätte sich Linux kaum über die Grenzen Finnlands hinaus verbreitet. Wie in vielen anderen Bereichen von Wissenschaft und Technik ist Englisch die Umgangssprache der internationalen Computergemeinde. Obwohl Linux in C geschrieben ist, kann es nur deshalb international entwickelt werden, weil sich der C-Quelltext englisch lesen läßt: Funktionsnamen, Kommentare und Meldungen sind in dieser Sprache abgefaßt.

Was für die Entwicklung eines Betriebssystems sinnvoll ist, muß aber noch lange nicht für dessen Anwendung gut sein. Auch wenn der erforderliche englische Wortschatz für den Umgang mit Linux sehr klein ist, würde die Beschränkung auf einen bestimmten nationalen Stil den Gebrauchswert von Linux mindern. Es kann schnell zu schwerwiegenden Mißverständnissen kommen, wenn der in Linux verwendete Stil vom nationalen abweicht.

Beispielsweise ist die Zeichenkette `04/11/1994' wegen der Jahreszahl 1994 leicht in die Kategorie Datum einzuordnen. In Deutschland ist die Verwendung eines / als Trennzeichen zwischen den Teilen des Datums zwar unüblich, wegen der gebräuchlichen Abfolge Tag-Monat-Jahr wird das Beispiel aber leicht als 4. November 1994 gelesen. In den USA bezeichnet dieses Datum eindeutig den 11. April 1994.

Um diesem Problem bereits auf Betriebssystemebene angemessen zu begegnen, unterstützt Linux (oder besser: die Standard-C-Bibliothek) sogenannte Locales und bietet in gewissem Umfang ``Native Language Support".

- 
- [Überblick](#)
  - [Anwendung von Locales](#)
  - [Erzeugung und Installation der Regelsätze](#)
  - [Native Language Support \(NLS\)](#)



**Next:** [Anwendung von Locales](#) **Up:** [Locales und Native Language](#) **Previous:** [Locales und Native Language](#)

# Überblick

Mit dem Ziel, Programme an nationale Besonderheiten anpassen zu können, ohne dabei die Portierbarkeit eines Programms einzuschränken, sind im POSIX-Standard sogenannte **Locales** beschrieben. Mit Locales kann das oben dargestellte Datumsproblem gelöst werden, und es können die nationalen Zahlen- und Währungskonventionen sowie die lexikografischen Besonderheiten der Zeichensätze behandelt werden.

Seit der Standard-C-Library-Version 4.6.20 vom November 1993 unterstützt Linux 7 verschiedene Kategorien von Locales:

## LC\_COLLATE

In der Kategorie LC\_COLLATE werden die Regeln zur lexikografischen Ordnung der nationalen Zeichensätze angepaßt. Durch LC\_COLLATE kann also die Sortierung von Zeichenketten verändert werden.

## LC\_CTYPE

In der Kategorie LC\_CTYPE wird die Zuordnung der Schriftzeichen zu Typen wie Ziffer, Buchstabe, Satzzeichen usw. geregelt. Damit können beispielsweise die deutschen Umlaute zu normalen Buchstaben erklärt werden, bei denen auch die Umwandlung von Groß- und Kleinbuchstaben korrekt durchgeführt wird.

## LC\_MONETARY

Die Kategorie LC\_MONETARY regelt die Formatierung von Zahlen bei der Darstellung von Geldbeträgen. Zu den veränderbaren Formatbestandteilen gehören neben dem nationalen und dem internationalen Währungssymbol die Darstellung von positiven und negativen Vorzeichen und die Symbolik zur Gruppierung der Ziffern.

## LC\_NUMERIC

Die Kategorie LC\_NUMERIC behandelt die Symbolik zur Gruppierung der Ziffern einer beliebigen Zahl. Hier kann beispielsweise das Trennzeichen zwischen dem Ganzzahl- und dem Dezimalteil einer Zahl geändert werden.

## LC\_TIME

In der Kategorie LC\_TIME wird das Datumsformat bestimmt.

## LC\_MESSAGES

Das Locale LC\_MESSAGES erlaubt die Anpassung aller Textmeldungen an die jeweilige Landessprache. Die Verwendung dieses Locale weicht von POSIX ab und ist im X/Open Portability Guide beschrieben. Diese Funktionalität wird auch als Native Language Support (NLS) bezeichnet.

## LC\_RESPONSE

Die Kategorie LC\_RESPONSE regelt unter Linux die landessprachlichen Varianten der zustimmenden bzw. ablehnenden Antwort. Diese Funktion wird in POSIX eigentlich LC\_MESSAGES zugeordnet. In zukünftigen Versionen der Library ist deshalb mit dem

Verschwinden dieses Locale zu rechnen.

Die Regeln für die verschiedenen Kategorien von Locales werden zur Laufzeit aus bestimmten Dateien gelesen. Indem diese Dateien geändert werden, können automatisch alle Programme, die mit Locales arbeiten, auf eine neue Umgebung angepaßt werden.

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [Anwendung von Locales](#) **Up:** [Locales und Native Language](#) **Previous:** [Locales und Native Language](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

# Anwendung von Locales


Weil die Locales bereits auf der Ebene der Standard-C-Library von einigen wichtigen Funktionen benutzt werden, können prinzipiell alle Programme, die solche Funktionen benutzen, in den oben genannten Kategorien an die jeweiligen nationalen Konventionen angepaßt werden.

Außerdem stehen Locales auch jedem Anwenderprogramm über bestimmte Datenstrukturen direkt zur Verfügung. Damit ist es möglich, ein Programm auch über den von den Bibliotheksfunktionen abgedeckten Bereich hinaus an nationale Konventionen anzupassen.

Damit bestimmte nationale Locales wirksam werden, müssen zuerst die Locales allgemein aktiviert und dann ein spezielles Locale durch entsprechende Umgebungsvariablen ausgewählt werden.

Die Aktivierung der Locales muß in jedem Fall durch das Programm selbst geschehen. Das heißt, auch wenn ein Programm mit Funktionen aus der Standard-C-Bibliothek arbeitet, die Locales unterstützen, muß innerhalb des Programms durch einen Aufruf der Funktion `setlocale(3)` die Verwendung der Locales ausdrücklich angefordert werden. Wenn die Locales nicht aktiviert sind, arbeiten die Bibliotheksfunktionen nach den vom POSIX-Standard definierten internationalen Regeln für C-Programme.

Ob ein bestimmtes Binärprogramm mit Locales arbeitet oder nicht, läßt sich nur durch einen Versuch ermitteln. Bei allen Programmen, zu denen Sie die Sourcen haben, können Sie natürlich die Verwendung von Locales durch entsprechende Ergänzungen im Quelltext aktivieren.

Wenn ein Programm Locales unterstützt und aktiviert, erfolgt die Auswahl eines bestimmten nationalen Regelsatzes für eine oder mehr Kategorien normalerweise  durch bestimmte Umgebungsvariablen. Neben den oben genannten Bezeichnungen der 7 verschiedenen Kategorien können noch die Variablen `LC_ALL` und `LANG` zur Bestimmung der Regelsätze beitragen.

Die Bezeichnung für einen bestimmten nationalen Regelsatz ist installationsabhängig. Im POSIX-Standard sind lediglich die beiden Namen `C` und `POSIX` gleichbedeutend für den Standardregelsatz ohne nationale Anpassungen fest vorgeschrieben. In der Praxis werden die Namen für die nationalen Regelsätze aus drei Teilen zusammengesetzt. Der erste Teil bezeichnet die Sprache, der zweite das Territorium und der dritte die Zeichencodierung.

Die Namen aller auf Ihrem System installierten Locales erfahren Sie vom `locale`-Kommando: 

```
$ locale -a
C
ISO-8859-1
de_AT.88591
de_BE.88591
de_CH.88591
de_DE.88591
de_LU.88591
it_IT.88591
```

```
fr_BE.88591
fr_CA.88591
fr_CH.88591
fr_FR.88591
fr_LU.88591
$ _
```

Beispielsweise ist `de_CH.88591` das Locale für die deutschsprachige Schweiz mit Verwendung der Latin-1-Zeichencodierung, `de_DE.88591` ist das entsprechende Locale für die BRD.

Sie können alle Programme, die mit Locales der Kategorie `LC_TIME` arbeiten, auf den schweizerischen Regelsatz umstellen, indem Sie der Umgebungsvariablen `LC_TIME` den Wert `de_CH.88591` zuweisen. In der `bash` geschieht das durch das Shellkommando `export (1)`:

```
$ export LC_TIME=de_CH.88591
$ _
```

Um gleichzeitig einen Regelsatz für alle Kategorien zu bestimmen, kann der Name des gewünschten Satzes in der Umgebungsvariablen `LC_ALL` oder `LANG` gespeichert werden. Die Priorität von `LC_ALL` ist höher als die der einzelnen Kategorien `LC_TIME`, `LC_CTYPE` etc., die Priorität von `LANG` dagegen niedriger als alle anderen. Wenn `LC_ALL` gesetzt ist, werden die für einzelne Kategorien gesetzten Umgebungsvariablen also ignoriert; wenn `LANG` gesetzt ist, wird der darin bezeichnete Regelsatz nur verwendet, sofern er nicht durch eine der anderen Variablen näher bestimmt wird.

Wenn Sie das `locale`-Kommando ohne Schalter aufrufen, erhalten Sie eine Liste aller Kategorien mit den aktuellen Locales.

```
$ export LANG=de_DE.88591
$ export LC_TIME=de_CH.88591
$ export LC_MESSAGES=C
$ locale
LANG=de_DE.88591
LC_COLLATE="de_DE.88591"
LC_CTYPE="de_DE.88591"
LC_MONETARY="de_DE.88591"
LC_NUMERIC="de_DE.88591"
LC_TIME=de_CH.88591
LC_MESSAGES=C
LC_RESPONSE="de_DE.88591"
LC_ALL=
$ _
```

Die in Anführungszeichen eingeschlossenen Belegungen sind nicht explizit definiert, sondern aus anderen Umgebungsvariablen, hier `LANG`, abgeleitet.

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [Erzeugung und Installation der](#) **Up:** [Locales und Native Language](#) **Previous:** [Überblick](#)



# Erzeugung und Installation der Regelsätze

Außer dem Standardregelsatz C oder POSIX, der fest im Programmtext der Bibliotheksfunktionen verankert ist und nur in geringem Umfang verändert werden kann, müssen alle Locales als Regeldateien im Laufzeitsystem installiert werden.

Um bei älteren Linux-Distributionen Locales hinzuzufügen, müssen zusätzlich die Shared Libraries vor Version 4.6.20 durch aktuelle ersetzt werden.

Die Regeldateien enthalten Binärdaten und können nicht direkt mit einem Texteditor erzeugt werden. Sie entstehen, indem spezielle Textdateien von geeigneten Programmen weiterverarbeitet werden.


Im POSIX-2-Standard ist das Format von zwei Textdateien zur Erzeugung der Locales beschrieben. In der ersten Datei wird der Zeichensatz definiert, unter Linux ist das meistens ISO 8859-1 (Latin-1), die zweite Datei beschreibt Regeln für die sechs Locale-Kategorien.

Aus diesen Textdateien kann das Programm `localedef` die binären Regeldateien erzeugen. Die Regelsätze werden im Verzeichnis `/usr/lib/locale` installiert. Jeder Regelsatz befindet sich dort in einem eigenen Unterverzeichnis, dessen Name mit dem des Locale identisch ist.

Wenn Sie unter Linux mit der üblichen Latin-1-Zeichencodierung arbeiten, müssen Sie sich diese Mühe in der Regel nicht machen. Passende Locales für den deutschen Sprachraum und für andere europäische Länder hat Jochen Hein an der Universität von Clausthal-Zellerfeld zusammengestellt.

Sie finden die Pakete und alle in diesem Zusammenhang brauchbaren Tools auf dem FTP-Server `ftp.tu-clausthal.de:/pub/Linux/SLT/nls` und auf allen Servern, die dieses Verzeichnis spiegeln.

# Native Language Support (NLS)

Über die in POSIX-2 definierte Funktionalität für das Locale LC\_MESSAGES  hinaus ermöglicht Linux die Ausgabe von verschiedensten Programm- oder Systemmeldungen in unterschiedlichen Sprachen. Linux benutzt dazu die im X/Open Portability Guide beschriebene Methode, bei der alle Textmeldungen in sogenannten Katalogen im Laufzeitsystem außerhalb der Programmdatei gespeichert werden. Diese Kataloge können auch ohne Zugang zu den Quelltexten des Programms in jede Sprache übersetzt werden.

Die Standard-C-Bibliothek von Linux unterstützt Message-Kataloge. Mit den Quelltexten wird auch ein deutschsprachiger Katalog für alle Meldungen der Bibliotheksfunktionen ausgeliefert. Wenn dieser Katalog als `libc.cat` wie die Regeln der Locales in einem passenden Unterverzeichnis von `/usr/lib/locales` installiert wird, können diese anstelle der englischsprachigen Meldungen angezeigt werden.

```
$ ls /foo/bar
ls: /foo/bar: No such file or directory
$ export LANG=de_DE.88591
$ ls /foo/bar
ls: /foo/bar: Datei oder Verzeichnis nicht gefunden
$ _
```

Die Auswahl eines Katalogs erfolgt zur Laufzeit des Programms durch die Umgebungsvariable LC\_MESSAGES oder durch LC\_ALL bzw. LANG, wie bei den Locales oben beschrieben.

Der Native Language Support ist nicht auf die Bibliotheksfunktionen beschränkt. Die Free Software Foundation arbeitet daran, ihre Tools und Programme mit NLS auszustatten.

**Next:** [Was ist eigentlich die](#) **Up:** [Das Linux Anwenderhandbuch](#) **Previous:** [Native Language Support \(NLS\)](#)

# Literaturliste

## Linux

Dieses Handbuch kann trotz seines Umfangs nicht alle Fragen zu Linux und der damit verbreiteten Freien Software beantworten. Es gibt aber eine Reihe guter Texte auf elektronischen Datenträgern, die Sie sich auf dem Bildschirm durchlesen oder auf Papier ausdrucken können.

Zuerst sind die Texte des Linux-Dokumentation-Project zu nennen, die auf manchen Distributionen enthalten sind oder über das Internet bezogen werden können.

*Johnson, M.K. Linux Kernel Hacker's Guide* (Version 0.7)

*Kirch, O. Linux Network Administrator's Guide* (aktuelle Version 1.0)

*Welsh, M. Linux Installation and Getting Started* (Version 2.3)

*Wirzenius, L. Linux System Administrator's Guide* (Version 0.5 nur HTML)

*Greenfield, L. Linux User's Guide* (Version beta-1)

*Goldt, Sven The Linux Programmer's Guide* (Version 0.4)

Eine wichtige Quelle für aktuelle Information über Linux ist das USENET. Hier werden regelmäßig die HOWTOs gepostet, in denen häufig gestellte Fragen beantwortet und wertvolle Tips weitergegeben werden. Unter anderem gibt es HOWTOs zu folgenden Themen:

*Raymond, Eric Distribution HOWTO*

*Gortmaker, P. Ethernet HOWTO*

*FRiC Linux Hardware Compatibility HOWTO*

*Welsh, M. The Linux Installation HOWTO*

*Dawson, T. The Linux NET-2 HOWTO*

*Taylor, G. The Linux Printing HOWTO*

*Eckhardt, D. The Linux SCSI HOWTO*

Es gibt mittlerweile mehrere gute Linux-Bücher in deutscher Sprache.

*Hekman, J. & O'Reilly Linux in a Nutshell, Deutsche Ausgabe* 1997 463S. (O'Reilly)

*Kirch, Olaf Linux Wegweiser für Netzwerker* 1996 364 S. (O'Reilly)

*Kofler, Michael Linux Installation, Konfiguration, Anwendung* 2. Aufl. 1995 ca. 797S. (A-W)

*Beck, Michael u.a. Linux-Kernel-Programmierung* 3. Aufl 1995 504S. (Addison-Wesley)



*Welsh, M. Running Linux, 2nd Edition (englisch) 1996 650S. (O'Reilly)*

*Bailey, E.C. maximum rpm (englisch) 1997 442S. (RedHat Software)*

## **Unix allgemein**

*Gulbins, J., Obermayr, K. UNIX 4. Aufl. 1995 837S. (Springer)*

*Wolfinger, C. Keine Angst vor UNIX. Ein Lehrbuch für Einsteiger 1992 312S. (VDI)*

*Peek, J., O'Reilly, T., Loukides, M. UNIX Power Tools, 2nd Ed. 1997 1082S. (O'Reilly)*

*Libes, D., Ressler, S. Live with UNIX: A Guide for Everyone 1989 368S. (Prentice Hall)*

## **Unix intern**

*Bach, M.J. The Design of the UNIX Operating System 1986 270S. (Prentice Hall)*

*Goodheart, B. & Cox, J. UNIX SVR4 Reise durch den Zaubergarten 1994 796S. (Prentice Hall)*

*Vahalia, U. UNIX Internals, The New Frontiers 1996 601S. (Prentice Hall)*

*Lions, J., Salus, P. (Editor) Lions' Commentary on UNIX 6th Edition 1977 (Peer to Peer)*

*McKusik, Bostic, Karels, Quarterman The 4.4 BSD UNIX Operating System 1996 580S. (A-W)*

*Jolitz, W.F. & Jolitz, L. The Basic Kernel Source Code Secrets 1996 530S. (Peer-To-Peer)*

*Tanenbaum, A.S. Operating Systems. Design and Implementation 1987 768S. (PH)*

## **Unix Systemverwaltung**

*Frisch, A. UNIX Systemadministration 1996 776S. (O'Reilly) **sehr gut!***

*Nemeth, E., Snyder, Seebass Unix System Administration 1995 779S. (PH)*

*Richter, H. UNIX V.4 Systemverwaltung 1993 350S. (Addison-Wesley)*

## **X Window System**

*Gilly, D., O'Reilly, T. The X Window System in a Nutshell 1992 411S. (O'Reilly)*

*Jones, O. Introduction to the X Window System 1989. (Prentice-Hall)*

*Mansfield, N. The X Window System: A User's Guide 1992 400S. (Addison-Wesley)*

*Mansfield, N. The Joy of X 1993 350S. (Addison-Wesley)*

*Quercia, V., O'Reilly, T. X Window System User's Guide 1992 752S. (O'Reilly).*

## **Vernetzung**

*Hunt, C. TCP/IP Network Administration 1992 502S. (O'Reilly)*

*Rost, M., Schack M. Der Internet-Praktiker 1996 1008S. (Heise)*

*Liu, C., Albitz, P. DNS and BIND, 2nd Edition 1996 438S. (O'Reilly)*

*Costales, B., Allman, E., Rickert, N. Sendmail, 2nd Edition 1997 1050S. (O'Reilly)*

*Ravin, O'Reilly, Dougherty & Todino Using and Managing UUCP 1996 424S. (O'Reilly)*

*Rost, M., Schack M. Handbuch uucp-Networking 1997 319S. (Heise)*

## Programmierung

*Kernighan B., Ritchie, D. Programmieren in C 1988 ca280S. (Hanser)*

*Stevens, W.R. Programmierung in der UNIX-Umgebung 1995 804S. (A-W)*

*Stevens, W.R. UNIX Network Programming 1990 (Prentice Hall)*

- 
- [Linux](#)

---

Next	Up	Previous	Contents	Index
------	----	----------	----------	-------

**Next:** [Was ist eigentlich die](#) **Up:** [Das Linux Anwenderhandbuch](#) **Previous:** [Native Language Support \(NLS\)](#)

*Das Linux Anwenderhandbuch  
(C) 1997 [LunetIX](#)*

# Was ist eigentlich die GPL

Das Linux Betriebssystem, die meisten der in diesem Buch beschriebenen Programme und Teile des Handbuches selbst unterliegen der GNU General Public License (GPL). Mit dieser Lizenz machen die Urheber und Inhaber des Copyright ihr Produkt zu Freier Software. Das Wichtigste an der GPL ist die dahinter stehende Idee der Freien Software. Um dieser Idee auch in der wenig ideellen Welt des Softwaremarktes eine standfeste Position zu geben, hat Richard Stallman die Free Software Foundation gegründet und die General Public License herausgegeben.

Es gibt noch andere Lizenzbestimmungen, die ein Programm zu Freier Software machen. Die wichtigsten sind das ``Berkeley Copyright'', unter dem das Freie BSD und einige Programme für Linux veröffentlicht sind und das Copyright des X Consortium, unter dem das X Window System im allgemeinen und XFree86 im speziellen stehen. Diese Lizenzen versuchen nicht, wie die GPL, den Umgang mit Freier Software im Detail zu reglementieren. Insbesondere sind sie weniger strikt, was die Verwendung von Code in anderen Programmen angeht. Die Lizenztexte finden Sie bei den Sourcen aller Programme, deren Urheber sie unter diesen Bedingungen veröffentlicht haben.

Die General Public License ist die ausgefeiltste aller Lizenzen für Freie Software. Um Ihnen den Inhalt der GPL leichter verständlich zu machen, drucken wir hier einen im wesentlichen inhaltlich mit der GPL übereinstimmenden Text.

Als Grundlage konnten wir eine deutsche Übersetzung der GPL verwenden, die uns freundlicherweise von Katja Lachmann (na194@fim.uni-erlangen.de) zur Verfügung gestellt wurde. In die Bearbeitung sind wertvolle Anregungen von Ulf Möller (um@ulf.mali.sub.org) eingeflossen.

Bei diesem Text handelt es sich nicht um eine durch die Free Software Foundation bestätigte Übersetzung der General Public License. Ähnlichkeiten mit dem Original sind beabsichtigt, sollen aber nicht zu einer Verwechslung dieses Textes mit der GPL selbst führen.

Im Vorwort zur GPL werden die wesentlichen Punkte der Lizenz ohne die verbindlichen Formulierungen des eigentlichen Lizenztextes eingeführt.

- 
- [Das Wesentliche in Kürze](#)
  - [Die GPL im Einzelnen](#)

# Das Wesentliche in Kürze

Die Lizenzen für die meiste Software sollen verhindern, daß Sie die Programme weitergeben und verändern können. Im Gegensatz dazu will die GNU General Public License sicherstellen, daß freie Software von jedem benutzt und verändert werden kann - um zu gewährleisten, daß die Software für alle Benutzer frei ist. Die General Public License gilt für den Großteil der von der Free Software Foundation herausgegebenen Software und für alle anderen Programme, deren Autoren ihr Werk der GPL unterstellt haben. (Ein anderer Teil der Software der Free Software Foundation unterliegt stattdessen der GNU Library General Public License). Auch Sie können diese Möglichkeit der Lizenzierung für Ihre Programme anwenden.

Wenn in der GPL von "freier Software" gesprochen wird, ist wirklich Freiheit gemeint, nicht der Preis. Die General Public License soll sicherstellen, daß Sie die Freiheit haben, Kopien freier Software zu verbreiten (und etwas für diesen Service zu berechnen, wenn Sie möchten), daß Sie den Quellcode erhalten haben oder bekommen können, wenn Sie wollen, daß Sie die Software ändern oder Teile davon in neuen freien Programmen verwenden können, und daß Sie wissen, daß Sie dies alles tun dürfen.

Damit Ihre Rechte geschützt sind, muß die Lizenz Einschränkungen machen, die es jedem verbieten, Ihnen diese Rechte zu verweigern oder Sie aufzufordern, auf diese Rechte zu verzichten. Aus diesen Einschränkungen folgen bestimmte Verantwortlichkeiten für Sie, wenn Sie Kopien der Software verbreiten oder sie verändern.

Beispielsweise müssen Sie den Empfängern alle Rechte gewähren, die Sie selbst haben, wenn Sie - kostenlos oder gegen Bezahlung - Kopien eines solchen Programmes verbreiten. Sie müssen sicherstellen, daß auch sie den Quellcode erhalten haben bzw. bekommen können. Und Sie müssen ihnen diese Bedingungen zeigen, damit sie ihre Rechte kennen.

Die Free Software Foundation und die GPL schützen Ihre Rechte in zwei Schritten: (1) sie stellen die Software unter ein Copyright und (2) sie bieten Ihnen die General Public License an, die Ihnen das Recht gibt, die Software zu vervielfältigen, zu verbreiten und/oder zu verändern.

Um die Autoren und sich selbst zu schützen, will die FSF sicherstellen, daß jeder erfährt, daß für diese freie Software keine Garantie besteht. Wenn die Software von jemand anderem modifiziert und weitergegeben wird, möchte die FSF, daß die Empfänger wissen, daß sie nicht das Original erhalten haben, damit von anderen verursachte Probleme nicht die Reputation des ursprünglichen Autors schädigen.

Schließlich ist jedes freie Programm permanent durch Software-Patente bedroht. Die FSF möchte die Gefahr ausschließen, daß Distributoren eines freien Programmes individuell Patente auf ein Programm erhalten, mit dem Ergebnis, daß das Programm proprietär wird. Um dies zu verhindern, hat sie klar gemacht, daß jedes Patent für freie Benutzung durch jedermann lizenziert werden muß oder überhaupt nicht lizenziert werden darf.

**Next:** [Die GPL im Einzelnen](#) **Up:** [Was ist eigentlich die](#) **Previous:** [Was ist eigentlich die](#)

*Das Linux Anwenderhandbuch*

(C) 1997 [LunetIX](#)

# Die GPL im Einzelnen

Im eigentlichen Lizenztext stehen die genauen Bedingungen für die Vervielfältigung, Verbreitung und Bearbeitung:

0.

Die General Public License (GPL) gilt für jedes Programm und jedes andere Werk, in dem ein entsprechender Vermerk des Urhebers darauf hinweist, daß das Werk unter den Bestimmungen der General Public License verbreitet werden darf. Im folgenden wird jedes derartige Programm oder Werk als „das Programm“ bezeichnet.

Als „auf dem Programm basierendes Werk“ wird das Programm sowie jegliche Bearbeitung im Sinne des Urheberrechts bezeichnet; das bedeutet, ein Werk, das das Programm, auch auszugsweise, unverändert oder verändert, und/oder in eine andere (Compiler-) Sprache übersetzt, enthält. (Im folgenden wird die Übersetzung ohne Einschränkung in „Bearbeitung“ eingeschlossen.) Jeder Lizenznehmer wird im Lizenztext als „Sie“ angesprochen.

Andere Handlungen als Vervielfältigung, Verbreitung und Bearbeitung werden von der General Public License nicht berührt; sie fallen nicht in ihren Anwendungsbereich. Der Vorgang der Ausführung des Programmes wird nicht eingeschränkt, und die Ausgabe des Programmes unterliegt der Lizenz nur, wenn der Inhalt ein auf dem Programm basierendes Werk darstellt (unabhängig davon, daß die Ausgabe durch die Ausführung des Programmes erfolgte). Ob dies zutrifft, hängt davon ab, was das Programm tut.

1.

Sie dürfen auf beliebigen Medien unveränderte Kopien des Quellcodes vom Programm, wie Sie ihn erhalten haben, anfertigen und verbreiten. Voraussetzung hierfür ist, daß Sie mit jeder Kopie einen entsprechenden Copyright-Vermerk, sowie einen Haftungsausschluß veröffentlichen. Sie müssen alle Vermerke, die sich auf die Lizenz und das Fehlen einer Garantie beziehen, unverändert lassen. Des weiteren müssen Sie allen anderen Empfängern des Programmes zusammen mit dem Programm eine Kopie der GPL geben.

Sie dürfen für den eigentlichen Kopiervorgang eine Gebühr verlangen, und es steht Ihnen frei, gegen Entgelt eine Garantie für das Programm anzubieten.

2.

Sie dürfen Ihre Kopie(n) des Programmes oder einen Teil davon verändern, wodurch ein auf dem Programm basierendes Werk entsteht; Sie dürfen derartige Bearbeitungen unter den Bestimmungen des Abschnitts 1 vervielfältigen und verbreiten, vorausgesetzt, daß zusätzlich alle folgenden Bedingungen erfüllt werden:

(a)

Sie müssen die veränderten Dateien mit einem auffälligen Vermerk versehen, der auf die von Ihnen vorgenommene Modifizierung und das Datum jeder Änderung hinweist.

(b)

Sie müssen dafür sorgen, daß jede von Ihnen verbreitete oder veröffentlichte Arbeit, die ganz oder teilweise von einem freien Programm oder Teilen davon abgeleitet ist, Dritten gegenüber als Ganzes unter den Bedingungen der GPL ohne Lizenzgebühren zur Verfügung gestellt wird.

(c)

Wenn das veränderte Programm normalerweise beim Lauf interaktiv Kommandos einliest, müssen Sie dafür sorgen, daß es, wenn es auf dem üblichsten Wege für solche interaktive Nutzung gestartet wird, eine Meldung ausgibt oder ausdruckt, die einen geeigneten Copyright-Vermerk enthält sowie einen Hinweis, daß es keine Gewährleistung gibt (oder anderenfalls, daß Sie Garantie leisten) und daß die Benutzer das Programm unter diesen Bedingungen weiter verbreiten dürfen. Auch muß der Benutzer darauf hingewiesen werden, wie er eine Kopie der GPL ansehen kann. (Ausnahme: Wenn das Programm selbst interaktiv arbeitet, aber normalerweise keine derartige Meldung ausgibt, muß Ihr auf dem Programm basierendes Werk auch keine solche Meldung ausgeben).

Diese Anforderungen betreffen das veränderte Werk als Ganzes. Wenn identifizierbare Teile des Werkes nicht von dem Programm abgeleitet sind und vernünftigerweise selbst als unabhängige und eigenständige Werke betrachtet werden können, dann erstrecken sich die General Public License und ihre Bedingungen nicht auf diese Teile, sofern sie als eigenständige Werke verbreitet werden. Wenn Sie jedoch dieselben Teile als Teil eines Ganzen verbreiten, das ein auf dem Programm basierendes Werk darstellt, dann muß die Verbreitung des Ganzen nach den Bedingungen der GPL erfolgen. Hierbei werden die Rechte weiterer Lizenznehmer auf die Gesamtheit ausgedehnt, und damit auf jeden einzelnen Teil - unabhängig von der Person des Verfassers.

Es ist nicht der Zweck dieses Absatzes, Rechte für Werke zu beanspruchen oder Ihre Rechte an Werken zu bestreiten, die komplett von Ihnen geschrieben wurden; vielmehr ist es die Absicht der GPL, die Rechte zur Kontrolle der Verbreitung von Werken, die auf einem freien Programm basieren oder unter seiner auszugsweisen Verwendung zusammengestellt worden sind, auszuüben.

Die einfache Zusammenstellung eines anderen Werkes, das nicht auf dem freien Programm basiert, gemeinsam mit dem Programm oder einem auf dem Programm basierenden Werk, auf einem Speicher- oder Vertriebsmedium, fällt nicht in den Anwendungsbereich der GPL.

3.

Sie dürfen das Programm (oder ein darauf basierendes Werk wie in Abschnitt 2) als Objectcode oder in ausführbarer Form unter den Bedingungen von Abschnitt 1 und 2 vervielfältigen und verbreiten - vorausgesetzt, daß Sie dabei das folgende tun:

(a)

Liefern Sie zusätzlich den vollständigen, zugehörigen, maschinenlesbaren Quellcode auf einem Medium, das üblicherweise für den Datenaustausch verwendet wird, wobei die Verteilung unter den Bedingungen der Abschnitte 1 und 2 erfolgen muß; oder

(b)

Liefern Sie das Programm mit dem mindestens drei Jahre gültigen schriftlichen Angebot, jedem Dritten eine vollständige, maschinenlesbare Kopie des Quellcodes zu einem Preis, der die Kosten für die materielle Durchführung der Verteilung nicht übersteigt, zur Verfügung zu stellen. Der Quellcode muß unter den Bedingungen der Abschnitte 1 und 2

auf einem für den Datenaustausch üblichen Medium verbreitet werden; oder

(c)

Liefern Sie das Programm mit der Information, die auch Sie als Angebot zur Verteilung des korrespondierenden Quellcodes erhalten haben. (Diese Alternative gilt nur für nicht-kommerzielle Verbreitung und nur, wenn Sie das Programm als Objectcode oder in ausführbarer Form mit einem entsprechenden Angebot nach Unterabschnitt b erhalten haben.)

Unter Quellcode eines Werkes wird die Form des Werkes verstanden, die für Bearbeitungen vorzugsweise verwendet wird. Für ein ausführbares Programm bedeutet vollständiger Quellcode: der gesamte Quelltext aller Module, die das Programm beinhaltet, zusätzlich alle zugehörigen Schnittstellendefinitionen, sowie die Scripte, die die Kompilierung und Installation des ausführbaren Programmes kontrollieren. Als besondere Ausnahme braucht der verteilte Quellcode nichts zu enthalten, was normalerweise (entweder als Quellcode oder in binärer Form) mit den Hauptkomponenten des Betriebssystems (Kernel, Compiler usw.) verteilt wird, unter dem das Programm läuft - es sei denn, diese Komponente gehört zum ausführbaren Programm.

Wenn die Verbreitung eines ausführbaren Programmes oder des Objectcodes erfolgt, indem der Kopierzugriff auf eine dafür vorgesehene Stelle gewährt wird, so gilt die Gewährung eines gleichwertigen Zugriffs auf den Quellcode als Verbreitung des Quellcodes, auch wenn Dritte nicht gezwungen sind, die Quellen zusammen mit dem Objectcode zu kopieren.

4.

Sie dürfen das freie Programm nicht vervielfältigen, verändern, weiter lizenzieren oder verbreiten, sofern es durch die General Public License nicht ausdrücklich gestattet ist. Jeder anderweitige Versuch der Vervielfältigung, Modifizierung, Weiterlizenzierung und Verbreitung ist nichtig und beendet automatisch Ihre Rechte unter der GPL. Jedoch werden die Lizenzen Dritter, die von Ihnen Kopien oder Rechte unter der GPL erhalten haben, nicht beendet, solange diese die Lizenz voll anerkennen und befolgen.

5.

Sie sind nicht verpflichtet, die General Public License anzunehmen, da Sie sie nicht unterzeichnet haben. Jedoch gibt Ihnen nichts anderes die Erlaubnis, das Programm oder von ihm abgeleitete Werke zu verändern oder zu verbreiten. Diese Handlungen sind gesetzlich verboten, wenn Sie die Lizenz nicht anerkennen. Indem Sie das Programm (oder ein darauf basierendes Werk) verändern oder verbreiten, erklären Sie Ihr Einverständnis mit der General Public License und mit allen ihren Bedingungen bezüglich der Vervielfältigung, Verbreitung und Veränderung des Programms oder eines darauf basierenden Werkes.

6.

Jedesmal, wenn Sie das Programm (oder ein auf dem Programm basierendes Werk) weitergeben, erhält der Empfänger automatisch vom ursprünglichen Lizenzgeber die Lizenz, das Programm entsprechend den in der GPL festgelegten Bestimmungen zu vervielfältigen, zu verbreiten und zu verändern. Sie dürfen keine weiteren Einschränkungen für die Ausübung der darin zugestandenen Rechte des Empfängers vornehmen. Sie sind nicht dafür verantwortlich, die Einhaltung der Lizenz durch Dritte durchzusetzen.

7.

Sollten Ihnen infolge eines Gerichtsurteils, des Vorwurfs einer Patentverletzung oder aus einem anderen Grunde (nicht auf Patentfragen begrenzt) Bedingungen (durch Gerichtsbeschluß,



Vergleich oder anderweitig) auferlegt werden, die den Bedingungen der General Public License widersprechen, so befreien Sie diese Umstände nicht von den Bestimmungen in der GPL. Wenn es Ihnen nicht möglich ist, das Programm unter gleichzeitiger Beachtung der Bedingungen in der GPL und Ihrer anderweitigen Verpflichtungen zu verbreiten, dann können Sie als Folge das Programm überhaupt nicht verbreiten. Wenn zum Beispiel ein Patent nicht die patentgebührenfreie Weiterverbreitung des Programmes durch diejenigen erlaubt, die das Programm direkt oder indirekt von Ihnen erhalten haben, dann besteht der einzige Weg, das Patent und diese Lizenz zu befolgen, darin, ganz auf die Verbreitung des Programmes zu verzichten.

Sollte sich ein Teil dieses Abschnitts als ungültig oder unter bestimmten Umständen nicht durchsetzbar erweisen, so soll dieser Abschnitt seinem Sinne nach angewandt werden; im übrigen soll dieser Abschnitt als Ganzes gelten.

Zweck dieses Abschnittes ist nicht, Sie dazu zu bringen, irgendwelche Patente oder andere Eigentumsansprüche zu verletzen oder die Gültigkeit solcher Ansprüche zu bestreiten; dieser Abschnitt hat einzig den Zweck, die Integrität des Verbreitungssystems der freien Software zu schützen, das durch die Praxis öffentlicher Lizenzen verwirklicht wird. Viele Leute haben großzügige Beiträge zum weiten Bereich der mit diesem System verbreiteten Software im Vertrauen auf die konsistente Anwendung dieses Systems geleistet. Es liegt am Autor/Geber, zu entscheiden, ob er die Software mittels irgendeines anderen Systems verbreiten will; ein Lizenznehmer hat auf diese Entscheidung keinen Einfluß.

Dieser Abschnitt ist dazu gedacht, deutlich klarzumachen, was als Konsequenz aus dem Rest der General Public License betrachtet wird.

8.

Wenn die Verbreitung und/oder die Benutzung des Programmes in bestimmten Staaten entweder durch Patente oder durch urheberrechtlich geschützte Schnittstellen eingeschränkt ist, kann der Urheberrechtsinhaber, der das Programm unter die GPL gestellt hat, eine explizite geographische Begrenzung der Verbreitung angeben, indem diese Staaten ausgeschlossen werden, so daß die Verbreitung nur innerhalb und zwischen den Staaten erlaubt ist, die nicht ausgeschlossen sind. In einem solchen Fall beinhaltet die GPL die Beschränkung, als wäre sie im Lizenztext niedergeschrieben.

9.

Die Free Software Foundation kann von Zeit zu Zeit überarbeitete und/oder neue Versionen der General Public License veröffentlichen. Solche neuen Versionen werden vom Grundprinzip her der gegenwärtigen entsprechen, können aber im Detail abweichen, um neuen Problemen und Anforderungen gerecht zu werden.

Jede Version der Lizenz hat eine eindeutig unterschiedliche Versionsnummer. Wenn das Programm angibt, welcher Version und „any later version“ es unterliegt, so haben Sie die Wahl, entweder den Bestimmungen dieser Version zu folgen oder denen jeder beliebigen späteren Version, die von der Free Software Foundation veröffentlicht wurde. Wenn das Programm keine Versionsnummer angibt, können Sie eine beliebige Version wählen, die je von der Free Software Foundation veröffentlicht wurde.

10.

Wenn Sie den Wunsch haben, Teile des Programmes in anderen freien Programmen zu verwenden, deren Bedingungen für die Verbreitung anders sind, schreiben Sie an den Autor, um ihn um die Erlaubnis zu bitten. Für Software, die unter dem Copyright der Free Software

Foundation steht, schreiben Sie an die Free Software Foundation; die FSF macht zu diesem Zweck manchmal Ausnahmen. Die Entscheidung darüber wird von den beiden folgenden Zielen geleitet: dem Erhalten des freien Status von allen abgeleiteten Werken der freien Software und der Förderung der Verbreitung und Nutzung von Software generell.

## KEINE GEWÄHRLEISTUNG

11.

**Da das Programm ohne jegliche Kosten lizenziert wird, besteht keinerlei Gewährleistung für das Programm, soweit dies gesetzlich zulässig ist. Sofern nicht anderweitig schriftlich bestätigt, stellen die Urheber und/oder Dritte das Programm so zur Verfügung, „wie es ist“, ohne irgendeine Gewährleistung, weder ausdrücklich noch implizit, einschließlich, aber nicht begrenzt auf die Tauglichkeit und Verwendbarkeit für einen bestimmten Zweck. Das volle Risiko bezüglich Qualität und Leistungsfähigkeit des Programmes liegt bei Ihnen. Sollte das Programm fehlerhaft sein, übernehmen Sie die Kosten für notwendigen Service, Reparatur oder Korrektur.**

12.

**In keinem Fall, außer durch geltendes Recht gefordert oder schriftlich zugesichert, ist irgendein Urheber oder irgendein Dritter, der das Programm wie oben erlaubt modifiziert oder verbreitet hat, Ihnen gegenüber für irgendwelche Schäden haftbar, einschließlich jeglicher genereller, spezieller, zufälliger oder Folgeschäden, die aus der Benutzung des Programmes oder der Unbenutzbarkeit des Programmes folgen (einschließlich, aber nicht beschränkt auf Datenverluste, fehlerhafte Verarbeitung von Daten, Verluste, die von Ihnen oder einem Dritten erlitten werden, oder einem Versagen des Programms bei der Zusammenarbeit mit irgendeinem anderen Programm), selbst wenn ein Urheber oder Dritter über die Möglichkeit solcher Schäden unterrichtet worden war.**

Die General Public License schließt mit einer Anleitung, wie Sie eigene Werke unter die GPL stellen können.

---

<a href="#">Next</a>	<a href="#">Up</a>	<a href="#">Previous</a>	<a href="#">Contents</a>	<a href="#">Index</a>
----------------------	--------------------	--------------------------	--------------------------	-----------------------

**Next:** [Bestellung des Linux Anwenderhandbuchs](#) **Up:** [Was ist eigentlich die](#) **Previous:** [Das Wesentliche in Kürze](#)

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

**Next:** [Über dieses Dokument ...](#) **Up:** [Das Linux Anwenderhandbuch](#) **Previous:** [Bestellung des Linux Anwenderhandbuches](#)

# Index

[~](#)

[.emacs](#)

[.inputrc](#)

[.tcshrc](#)

[:-Shellkommando](#)

[^ \(CONTROL\)](#)

[^ \](#)

[^ ?](#)

[^ C](#)

[^ D](#)

[^ N](#)

[^ O](#)

[^ Q](#)

[^ R](#)

[^ S](#)

[^ V](#)

[^ W](#)

[^ Z , !\[\]\(aff7c69c44a5e015f18c35867ef3f5c3\_img.jpg\)](#)

[\[ \]](#)

[a.out , !\[\]\(1ed10657a19f9137278430c48fd18626\_img.jpg\) , !\[\]\(4dcc4b4891182a8bcced7bb00d8b5889\_img.jpg\)](#)

[ACK](#)

[active Datei \(News\)](#)

[adjtime](#)

[adjtime| \(](#)

[adjtime| \)](#)

[agetty](#)

[alias-Shellkommando , !\[\]\(fed825e7856867ee486f6761f9a89d91\_img.jpg\)](#)

[Aliases|seeSynonyme](#)

## Anfuhrungszeichen@Anführungszeichen

### Anhalten

eines Programms ,  , 

### ar

Arbeitsspeicher , 

### Arbeitsspeicher

mehr als 64MB

Archivieren von Daten , 

Artikel , 

### ATAPI

### atime

Ausgabeumleitung ,  , 

### autoconf

Backslash|seeFluchtsymbol

Backup|seeDatensicherung

Bandlaufwerke|seeMagnetbandlaufwerke

### bang path

basename ,  , 

basename|textbf

### bash

Arithmetik

Array-Variable

Aufruf

Aufrufoptionen

beenden , 

Eingabeaufforderung

Grammatik

interaktiv

Kontrollstrukturen|

Kontrollstrukturen|)

Menues@Menüs

Parameter

Parameter|

Parameter|)

[Shellkommandos|\(](#)

[Shellkommandos|\)](#)

[Shellscript](#)

[Signale ,](#) 

[Skriptfunktionen](#)

[Sonderzeichen](#)

[Variable|\(](#)

[Variable|\)](#)

[bash|\(](#)

[bash|\)](#)

[Batch ,](#) 

[Baudrate](#)

[bdflush](#)

**Beenden**

[einer Shell@Shell](#)

[Programm ,](#) 

[von Linux@Linux](#)

[Bell Laboratories](#)

[Benutzer](#)

**Benutzer**

[Authentifizierung|\(](#)

[Authentifizierung|\)](#)

[eintragen|\(](#)

[eintragen|\)](#)

[Gruppe wechseln ,](#) 

[ID ,](#) 

[Name](#)

[Benutzergruppen ,](#)  [, ,](#)  [, ,](#)  [, ,](#) 

[Benutzer|\( ,](#) 

[Benutzer|\) ,](#) 

[bg-Shellkommando](#)

[BildschirmAusgabe speichern](#)

[Bildschirm|\(](#)

[Bildschirm|\)](#)

[bind-Shellkommando](#)

[bind-Shellkommando|textbf](#)

[Binärdateien anzeigen](#) , 

[Binärkompatibilität|](#)

[Binärkompatibilität|\)](#)

[Binärpakete](#)

[Binärzahl](#)

[Bitmaps](#)

[Block](#)

[blockorientiertes Gerät](#)

[BogoMIPS](#)

[Bootdisk](#)

**Booten**

[Singleuser](#) , 

[Booten|](#)

[Booten|\)](#)

[Bootkonzept](#)

[Bootkonzept|](#)

[Bootkonzept|\)](#)

[Bootloader](#)

[Bootloader|](#)

[Bootloader|\)](#)

[Bootmanager](#) ,  , 

[Bootprompt](#) , 

[Bootprompt|](#)

[Bootprompt|\)](#)

[Bootselector](#)

[Bourne-Shell|seebash](#)

[BREAK](#)

[break-Shellkommando](#)

[builtin-Shellkommando](#)

[Busmäuse](#)

[Byte|seeZeichen](#)

[C News](#)

[Cache](#) ,  ,  ,  , 

[case Verzweigung](#)

[cat](#) ,  , 

[CD-ROM](#)

[CD-ROM](#)(  ,

[CD-ROM](#))  ,

[cd-Shellkommando](#) , 

[CDPATH Shellvariable](#) , 

[Character|seeZeichen](#)

[Chat-Skript](#)

[chgrp](#)

[chmod](#)(

[chmod](#))

[chown](#)

[chsh](#)

[cksum](#)

[clock](#)

[CMOS-Uhr \(Echtzeituhr\)](#) , 

[cmp](#)

[COFF](#)

[comm](#)

[command-Shellkommando](#)

[Compilierung](#)

[compress](#)

[conf.modules](#)

**Console**

[Bildschirm](#)(

[Bildschirm](#))

[reset](#)

[Tastatur](#)(

[Tastatur](#))

[Zeichensatz laden](#)

[Console](#)( 

[Console](#)) 

[continue-Shellkommando](#)


[cooked Modus](#)


[copy on write](#)

[core](#)

[cp](#) , 

[cpio](#)

[cpio|](#) ( , 

[cpio|](#) ) , 

[crashme](#)

[cron Dämon|](#)(

[cron Dämon|](#))

[crontab](#)

[crypt](#)

[csplit](#)

[ctime](#)

[cut](#)



[Dateiarten](#)

[Dateiattribute|](#) ( , 

[Dateiattribute|](#) ) , 

**Dateien**

[anlegen](#) , 

[anzeigen](#) ,  ,  ,  ,  ,  ,  , 

[archivieren](#) , 

[ausführbare](#)

[drucken](#) ,  , 

[drucken|](#)(

[drucken|](#))

[durchsuchen](#) , 

[komprimieren](#) , 

[konvertieren](#)

[kopieren](#) ,  ,  , 

[loeschen@löschen](#) ,  , 

[Namen](#) , 

[sortieren](#)



[sperr](#)[en](#)

[suchen](#)

[teilen](#) ,  , 

[Typ feststellen](#)

[umbenennen](#) , 

[verdoppeln](#)

[vergleichen](#) , 

[versteckte](#)

[Zeitmarken](#)

[Zugriffsrechte](#)

[Zugriffsrechte|](#)

[Zugriffsrechte|\)](#)

[zusammenfügen](#) ,  , 

## **Dateiname**

[absolut](#)

[relativ](#)

[Dateinamen Länge](#)

[Dateisystem einrichten|\)](#)

## **Dateisystem**

[einrichten](#)


[einrichten|](#)

[einrichten|\)](#)

[ext](#)

[ext2](#) , 

[ext2|](#) ( , 

[ext2|\)](#) , 

[fat](#)

[iso9660](#)

[iso9660|](#)

[iso9660|\)](#)

[Konzept](#)

[minix](#)

[minix|](#)

[minix|\)](#)

NFS




proc|

proc|)

pruefen@prüfen|( ,  , 

pruefen@prüfen|) ,  ,  , 

reparieren|( ,  , 

reparieren|) ,  ,  , 

Standard

Typen|

Typen|)

umsdos|

umsdos|)

vfs

xiafs , 

zusammenbauen

Dateisystem|( , 

Dateisystem|) , 

Datenfernübertragung|seeDFÜ

Datensicherung , 

**Datensicherung**

auf Disketten@Disketten , 

inkrementell , 

Methoden

Multivolume , 

Datensicherung|

Datensicherung|)

Datenströme

Datenzonen , 

**Datenzonen**

fragmentierte

date|

date|)

Datum , 

[Datum|](#)

[Datum|](#)

[DCF-77 Funkuhr](#)

[dd](#)

**Debian**

[Softwarepakete](#)

[declare-Shellkommando](#)

[demand loading](#)

[depmod](#)

[DES|](#)

[DES|](#)

[Devices|seeGerätedateien](#)

[Dezimalzahl](#)

[df](#)

**DFÜ**

[Akustikkoppler](#)

[asynchron](#)

[BAUD](#)

[Break](#)

[Carrier](#)

[Handshake](#)

[minicom|](#)

[minicom|](#)

[MNP1-5](#)

[Modem|](#)

[Modem|](#)

[Parität](#)

[Stopbits](#)

[synchron](#)

[TAE-Stecker](#)

[term|](#)

[term|](#)

[V42bis](#)

[Zeichenlänge](#)

[DFÜ|](#)

[DFÜ|](#)

[Diacriticals](#)

[Directory|see Verzeichnis](#)

[dirname](#) , 

[dirname|textbf](#)

[dirs-Shellkommando](#)

[dir|seels](#)

**Disketten**

[formatieren](#) , 

[Inhalt](#)

[kopieren](#)

[Laufwerke|](#)

[Laufwerke|\)](#)

[lesen](#)

[löschen](#)

[MS-DOS Dateisystem](#)

[Operationen](#)

[Verzeichnis anlegen](#)

**Verzeichnis**

[löschen](#)

[disown-Shellkommando](#)

**DOS**

[Dateisystem](#) ,  , 

[Disketten](#)

**dosemu**

[/etc/dosemu.conf@ /etc/dosemu.conf](#) , 

[Bootlaufwerk](#)

[Diskettenlaufwerke](#)

[DPMI](#)

[EMS](#)

[Festplattenlaufwerke](#)

[Optionen](#)

[serielle Schnittstellen](#)

[Tastatur-Konfiguration](#)

[Terminalunterstützung](#)

[Video-Konfiguration](#)

## X-Window-Unterstützung

### XMS

dosemu@dosemu

doshell

Doublequote|seeAnführungszeichen

Double|seeFließkommazahl

dpkg

DR-DOS

drucken ,  , 

**drucken**

interruptgesteuert

Postscript

drucken|

drucken|)

Druckerdämon|

Druckerdämon|)

Druckerfilter

Drucker|

Drucker|)

dselect

du

Dämonen|

Dämonen|)

e2fsck

echo

echo-Shellkommando

**Editor**

elvis (vi)|

elvis (vi)|)

in elm

sed|

sed|)

edit|seeelvis

efsck

egrep

**Eigentümer**

einer Datei

eines Prozesses

ändern

Eigentümer|(

Eigentümer|)

Eingabeaufforderung ,  , 

**Eingabeaufforderung**

sekundär

Eingabeumleitung , 

ELF ,  ,  , 

elm

**elm**

Editor

elm.rc

Konfiguration

Mail versenden

Mailbox

elvis|(

elvis|)

elvprsv

elvrec

emacs-Modus der *bash*|(

emacs-Modus der *bash*|)

enable-Shellkommando

Ende der Eingabe

env

Environment|seeProzeßumgebung

EOF

**Erweiterung**

Klammer

eval-Shellkommando

exec-Shellkommando

exit-Shellkommando

expand

[expire](#)

[export-Shellkommando](#)

[expr](#)

[ext2|seeDateisysteme](#)

[ex|seeelvis](#)

[fc-Shellkommando](#)

[fdformat](#)

[fdisk](#)

[fdprm Datei](#)

**Fehler**

[im EXT2 Dateisystem](#)

[im Linux-Kernel](#)

[in einem Softwarepaket](#)

[in einem Systemaufruf](#)

[in Programmen](#) . 

[Status danach](#)

**Festplatten**

[8-Bit \(XT\)](#)

[AT-Bus|](#)

[AT-Bus|\)](#)

[booten](#)

[booten|](#)

[booten|\)](#)

[Dateisystem einrichten|](#)

[Dateisystem einrichten|\)](#)

[freier Platz](#)

[größer 1024 Zylinder](#)

[partitionieren](#)

[partitionieren|](#)

[partitionieren|\)](#)

[SCSI](#)

[synchronisieren](#)

[verbrauchter Platz](#)

[Festplattencache](#)

[fg Shellkommando](#)

[FIFO](#) ,  , 

[file](#)

[File-System-Standard](#)

[File|seeDatei](#)

[Filter](#)

**Filter**

[für Drucker](#)

[find|](#)

[find|\)](#)

[fips](#)

[Fließkommazahl](#)

[Floppylaufwerke](#) , 

[Floppylaufwerke|](#)

[Floppylaufwerke|\)](#)

[Floppystreamer](#)

[Floppystreamer|](#)

[Floppystreamer|\)](#)

[Fluchtsymbol](#)

[fold](#)

[Font|seeZeichensatz](#)

[for-Schleifen](#)

[free](#)

[fsck Front-End](#)

[fsck.minix](#)

[fsck.xiafs](#)

[fstab Datei](#)

[fstab Datei|](#)

[fstab Datei|\)](#)

[ftape|seeFloppystreamer](#)

[FTP-Site](#)

[Funktionen im Shellscrip](#)

[Funktionsbibliothek](#)

[Gates](#)

[Gerätedateien](#) , 

[Gerätedateien|](#)



[Gerätedateien|](#)

[Gerätenummer|seeHauptgerätenummer](#)

[Gerätetreiber|](#)

[Gerätetreiber|\)](#)

[getopts Shellkommando](#) , 

[getty\\_ps](#)

[getty\\_ps|](#)

[getty\\_ps|\)](#)

[gettydefs Datei](#)

[getty|](#)

[getty|\)](#)

[GID|seeGruppen-ID](#)

[GMT](#)

[grep|](#)

[grep|\)](#)

[groff|](#)

[groff|\)](#)

[group Datei](#) ,  , 

[groups](#)

**Gruppe**

[ID](#)

[wechseln](#) , 

[ändern](#)

[Gruppe|](#)

[Gruppe|\)](#)

[gzip](#)

[halt](#)

[Harddisks|seeFestplatten](#)

[harter Link](#)

[hash Shellkommando](#)

[Hauptgerätenummer](#) , 

[HDB](#)

[head](#)

[Heimatverzeichnis](#) ,  ,  , 

[help-Shellkommando](#)

[Herunterfahren des Betriebssystems](#)

[Hexadezimalzahl](#)

[hexdump](#)

**Hilfe**

[Kommandos@zu Programmen](#)

[Shellkommandos@zu Shellkommandos  
zu Programmen](#)

[Hintergrundprozesse](#) ,  , 

**Hintergrundprozesse**

[anzeigen](#)

[beenden](#)

[starten](#) ,  , 

[von der Shell trennen](#)

[history-Shellkommando](#)

[History|](#)(

[History|](#))

[Hochkomma](#)

[HOME-Umgebungsvariable](#) ,  , 

[HoneyDanBer](#)

[hostname](#)

[hosts Datei](#)

[hosts.nntp](#)

[hpfs|seeDateisysteme](#)

[I-Nodes](#) ,  , 

[iBCS2|](#)(

[iBCS2|](#))

[id](#)

[if Verzweigung](#)

[IFS Shellvariable](#)

[indirekter Block](#)

[inews](#)

[initrd](#)

[inittab Datei](#)

[inittab Datei|](#)(

[inittab Datei|](#))

[init|](#)

[inkrementelles Backup](#)

[inn.conf](#)

[innd](#)

[INN|](#)

[INN|\)](#)

[input|seeelvis](#)

[insmod](#)

[insmod|textbf](#)

[install](#)

[Integer|seeDezimalzahl](#)

[Internet](#)

[IP-Adressen](#)

[iso9660|seeDateisysteme](#)

[issue Datei](#)

[Job Control](#)

[jobs-Shellkommando](#)

[join](#)

[Jokerzeichen](#) ,  , 

[kermit](#)

[Kernel](#) ,  , 

**Kernel**

[Crash](#)

[konfigurieren](#)

[Quelltexte entpacken](#)

[Speicher](#)

[kernel|](#)

[kernel|\)](#)

[Kernelmodule](#) ,  , 

**Kernelmodule**

[initialisieren](#)

[Symbole](#)

[Kernelmodule|](#)

[Kernelmodule|\)](#)

[Keyboard|seeTastatur](#)

[kill](#) , 

[Klammererweiterung](#)

**Kommandos**

[anhalten](#)

[im Hintergrund|](#)

[im Hintergrund|](#)

[mehrere in einer Zeile|](#)

[mehrere in einer Zeile|](#)

[verketten](#) , 

[Kommandosubstitution](#)

**Kommandozeile**

[automatisch erweitern](#)

[Editor|](#)

[Editor|](#)

[für den Kernel](#)

[History|](#)

[History|](#)

[Interpretation](#)

[Optionen](#) ,  , 

[Regeln|](#)

[Regeln|](#)

[Kommandozeileninterpreter|seebash](#)

[Kommandozeilenparameter](#)

[Kommandozeilenspeicher](#)

**Kommentare**

[in Shellscripts](#)

[Konfigurationsdateien](#)

[Konfigurationsdateien|](#)

[Konfigurationsdateien|](#)

**Kopieren**

[Dateien](#)

[Verzeichnisse](#)

[kswapd](#)

[ksyms](#)

[LC\\_ALL](#)

[ldconfig|](#)

[ldconfig|](#)

[less](#) , 

[let-Shellkommando](#)

[LILO](#)

[LILO|](#)

[LILO|](#)

[Link](#) ,  , 

**Link**

[auf ein Verzeichnis](#)

[symbolisch](#)

**symbolisch**

[schnell \(fast\)](#)

[Link|textbf](#)

[ln](#)

[LOADLIN](#)

[local-Shellkommando](#)

[Locales|](#)

[Locales|](#)

[Lockfiles](#)

**loeschen@löschen**

[einer Datei](#)

[eines Verzeichnisses](#)

[Login](#) , 

**Login**

[deslogin](#)

[S/Key](#)

[seriell|](#)

[seriell|](#)

[Shell](#) , 

[Login|](#)

[Login|](#)

[logname](#)

[logout-Shellkommando](#)

[loop-Device](#)

[lost+found](#) , 

[lpd|\(](#)

[lpd|\)](#)

[lpq](#)

[lpr](#)

[lprm](#)

[ls](#)

[ls|\(](#)

[ls|\)](#)

[magic Datei](#)

**Magnetband**

[mehrere Dateien](#)

[Multivolume](#)

[positionieren](#)

[Magnetbandlaufwerke](#)

[Magnetbandlaufwerke|\(](#) , 

[Magnetbandlaufwerke|\)](#) , 

[Mail](#)

**Mail**

[Adressen](#)

[Header](#)

[Routing](#)

[Software](#)

[testen](#)

[Major Device Number|seeHauptgerätenummer](#)

[MAKEDEV](#)

[man](#)

[Mandatory Locking](#)

[Manualpages](#) , 

**Manualpages**

[formatieren](#)

[Master Boot Record](#) , 

[Maus](#)

[mcopy](#)

[mdel](#)

[mdir](#)

[Menue in der Shell@Menü in der Shell](#)

[MET](#)

[Metamodus](#)

[Metataste](#)

[mformat](#)

[mgetty](#)

**[minicom](#)**

[Anwahl](#)

[Konfiguration](#)

[Login-Script](#)

[Modem-Device](#)

[Script-Sprache](#)

[minicom|](#)

[minicom|\)](#)

[Minor Device Number|seeUntergerätenummer](#)

[mkboot](#)

[mkdir](#) , 

[mke2fs](#)

[mkfifo](#)

[mkfs Front-End](#)

[mkfs.ext2](#)

[mkfs.minix](#)

[mkfs.xiafs](#)

[mknod](#)

[mksuper](#)

[mkswap](#)

[mkxfs](#)

[mmd](#)

[Modem](#)

[Modem|](#)

[Modem|\)](#)

[modprobe](#)

[modules|seeKernelmodule](#)

[more](#)

[more|](#)

[more|](#)

[motd Datei](#)

[mount](#)

[mount|](#)

[mount|](#)

[mrd](#)

[mread](#)

**MS-DOS**

[Dateisystem](#)

[mtime](#)

[mtools](#)

[mt|](#)

[mt|](#)

[Muelleimer@Mülleimer](#)

[Multimedia](#)

[Multitasking](#) , 

[Multisusersystem](#) , 

[Multisusersystem|](#)

[Multisusersystem|](#)

[Multivolume Archive](#) , 

[mv](#)

**Nachricht**

[an alle Benutzer](#)

[an einen Benutzer](#)

[NAK](#)

[named Pipe](#) , 

[Native Language Support|](#)

[Native Language Support|](#)

[Net-News](#)

[newgrp](#)

[newline@NEWLINE](#)

[newsfeeds](#)

[newsgroups](#)

[newsmaster](#)



[Newsspool](#)

[News|](#)

[News|\)](#)

[nice](#)

[nl](#)

[NLS|seeNative Language Support](#)

[nnrp.access](#)

[nnrpd](#)

[NNTP](#)

[nohup](#)

[nologin Datei](#)

[od](#)

[Oktalzahl](#)

[Optionen](#) , 

[Packer](#) , 

[Parametererweiterung](#)

[Parität](#)

**Partition**

[Größe](#)

[Partitionstabelle](#)

[Partitionstyp](#)

[Partition|](#)

[Partition|\)](#)

[passwd Datei](#)

[passwd Programm](#)

[paste](#)

[PATH-Umgebungsvariable](#) ,  ,  ,  , 

[pathalias](#)

[Paßwort](#) ,  , 

**Paßwort**

[Shadow-System|](#)

[Shadow-System|\)](#)

[Sicherheit|](#)

[Sicherheit|\)](#)

[Verschlüsselung](#)

[ändern](#)

[Permissions|seeZugriffsrechte](#)

[Pfadname](#)

[PGP|](#)

[PGP|\)](#)

[Pipelines](#) , 

[Polling](#) ,  , 

[popd-Shellkommando](#)

[Positionsparameter](#)

[PPP \(Point to Point Protocol\)](#)

[pr](#)

[printcap Datei](#)

[printenv](#)

[proc|seeDateisysteme](#)

[profile Datei](#)

[Prompt|seeEingabeaufforderung](#)

**Prozeß**

[abbrechen](#)

[beenden](#)

[Eigentum \(UID\)|](#)






[Eigentum \(UID\)|\)](#)

[Eltern-Kind Beziehung](#)

[Nummer](#)

[Status](#)

[Tabelle](#) ,  , 

[Umgebung](#) ,  ,  ,  ,  , 

[Pruefsummen@Prüfsummen](#) , 

[PS1 Shellvariable](#)

[psdatabase Datei](#)

[ps|](#)

[ps|\)](#)

[pushd-Shellkommando](#)

[pwd](#) ,  , 

**QIC**

[02 Bandlaufwerke , !\[\]\(cead67df4d82d6c83effe4f8699a7d8f\_img.jpg\)](#)

[117 Bandlaufwerke|\(](#)

[117 Bandlaufwerke|\)](#)

[Standards](#)

[Quote|seeHochkomma](#)

[Quotierung](#)

[Raid](#)

[RAM-Disk , !\[\]\(e474458956c9a37fbf9586ddb60a7fa1\_img.jpg\) , !\[\]\(4d1d3f2547aeece54bb6babd23f4121b\_img.jpg\)](#)

[RAM-Disk|\(](#)

[RAM-Disk|\)](#)

[raw Modus](#)

[rdev](#)

[read Shellkommando](#)

[readline , !\[\]\(b792654f2cef9719eabeb6c5be00811e\_img.jpg\)](#)

[readonly-Shellkommando](#)

[reboot](#)

[Rechnen in der Shell](#)

[reset der Console](#)

[resolver](#)

[return-Shellkommando](#)

[rm](#)

[rmdir , !\[\]\(c15650232aa6660c9deb34f3b82dcb72\_img.jpg\)](#)

[rmmod , !\[\]\(1ed10657a19f9137278430c48fd18626\_img.jpg\)](#)

[rnews](#)

[Rock-Ridge Erweiterung , !\[\]\(a25a22d88c5882f4a20f36103df86562\_img.jpg\)](#)

[root-Account , !\[\]\(06b7456efb47d301bca6298603e7f4fc\_img.jpg\) , !\[\]\(c1e9cd3169432c75af46916a2b923325\_img.jpg\)](#)

[Rootfilesystem , !\[\]\(2885535958616e9ec6b97903614c334b\_img.jpg\) , !\[\]\(9617090d30d69647517339fb4821ff48\_img.jpg\)](#)

[Rootpartition](#)

[rpm](#)

[rpm2cpio](#)

[rpm|\(](#)

[rpm|\)](#)

[RS-232](#)

[RTS/CTS](#)

[Rueckgabewert@Rückgabewert](#)

[Runlevel](#) , 

[Scheduler](#)

[Schuetzen von Sonderzeichen@Schützen von Sonderzeichen](#)

[Screendump](#)

**SCSI**

[Bandlaufwerke](#)

[Bandlaufwerke|](#)(

[Bandlaufwerke|](#))

[Hostadapter](#)

[Hostadapter|](#)(

[Hostadapter|](#))

[securetty Datei](#)

[sed|](#)(

[sed|](#))

[Sektor](#)

[select Menue@select Menü](#)

[serielle Schnittstelle](#) ,  , 

**serielle Schnittstelle**

[16550](#)

[FIFO](#)

[Multiport-Karten](#)

[serielle Schnittstelle|](#)(

[serielle Schnittstelle|](#))

[Session](#) , 

[set-Shellkommando](#)

[setfdprm|](#)(

[setfdprm|](#))

[Shared Libraries](#) ,  , 

[Shared Libraries|](#)(

[Shared Libraries|](#))

[Shell](#) ,  , 

[Shell-Level](#)

[Shellattribute verändern](#)

**Shellkommando**

[an/abschalten](#)

[aufrufen](#)

[shells Datei](#)

[Shellscript](#) ,  ,  ,  ,  ,  ,  ,  ,  ,  ,  ,  , 

**Shellscript**

[Argumente auswerten](#)

[Funktionen](#)

[Kommentare](#)

[Shellvariable](#)

**Shellvariable**

[einlesen](#)

[erzeugen](#)

[lokal](#)

[löschen](#)

[read only](#) ,  , 

[Shell|\(](#)

[Shell|\)](#)

[shift-Shellkommando](#)

[shopt-Shellkommando](#)

[shutdown](#) , 

[sh|seebash](#)

[Sicherheit|\(](#)

[Sicherheit|\)](#)

[Signale](#) , 

**Signale**

[abfangen](#)

[senden](#) ,  , 

[SIGINT](#)

[SIGSEGV](#) , 

[SIGTERM](#)

[Site](#)

[sleep](#)

[smail](#)

**smail**

[Installation](#)

[Konfigurationsdatei](#)

[Sockets](#)

**Softwarepakete**

[Abhängigkeit untereinander](#) , 

[aus Sourcen herstellen](#)

[Binär](#)

[Datenbank](#)

[Debian](#)

[Formate|\(](#)

[Formate|\)](#)

[Herstellung](#)

[Inhalt](#)

[Installation](#) , 

[Konfigurationsdateien](#)

[Löschen](#)

[Management](#)

[Management|\(](#)

[Management|\)](#)

[Prüfen](#)

[Reparatur](#)

[RPM-Format](#)

[Sourcen](#)

[Untersuchen](#)

[Upgrade](#) , 

[sort](#) , 

[sortieren](#)

[Sound](#)

[source-Shellkommando](#)

[Sourcepakete](#)

[Speicherplatz auf Platte](#)

[Spezialparameter](#)

[split](#)

[Spool-Verzeichnis](#)

[Standard-I/O](#)

[Standardausgabe](#)

**Standardausgabe**

[Verzweigen](#)

[Standardeingabe](#)

[Standardfehlerausgabe](#) , 

[Stapelverarbeitung](#)

[strace|](#)

[strace|](#)

[Streamer|seeMagnetbandlaufwerke](#)

[stty|](#)

[stty|](#)

[su](#)

**suchen**

[Ausdrücke in Dateien](#) , 

[eine Datei](#)

[Suchpfad](#) , 

[sum](#)

[Superblock](#)

[superformat|](#)

[superformat|](#)

[Superuser](#) ,  , 

[suspend-Shellkommando](#)

[swapping](#) , 

[symbolischer Link](#)

[sync](#)

[Synonyme](#) , 

[syslog Dämon|](#)

[syslog Dämon|](#)

[Systemaufruf](#)

**Systemmeldungen**

[protokollieren](#)

[Systemstart](#)

[Systemzeit](#)

[Systemzeit|](#)

[Systemzeit|\)](#)

[sysvinit](#)

**Tabulatoren**

[ersetzen](#)

[tac](#)

[tail](#)

[Tapes|seeMagnetbänder ,](#) 

[tar ,](#) 

[tar|](#)

[tar|\)](#)

**Tastatur**

[Funktionstasten](#)

[keycodes](#)

[nationale Belegung](#)

[Tabelle|](#)

[Tabelle|\)](#)

[Tastatur|](#)

[Tastatur|\)](#)

[TCP/IP](#)

[tee](#)

[teilen v. Dateien ,](#) 

[Telefonnetz](#)

**term**

[Konfiguration](#)

[Leitungstransparenz](#)

[linecheck](#)

[Server](#)

[termrc-Datei](#)

[tredir ,](#) 

[trsh ,](#) 

[tupload ,](#) 

[txconn ,](#) 

[Umgebungsvariablen](#)

[termcap Datei](#)



## Terminal

[einstellen](#)

[kontrollierendes](#)

[virtuell](#) , 

[virtuell|](#)(

[virtuell|](#))

[term|](#)(

[term|](#))

[test-Shellkommando](#)

[Texte formatieren](#)

[Tildenerweiterung](#)

[time](#) ,  , 

[times-Shellkommando](#)

[tin|](#)(

[tin|](#))

[Token](#) , 

[touch](#)

[trap-Shellkommando](#)

[tty](#)

[type-Shellkommando](#)

[UART](#)

[Uebersetzen eines C-Programms@Übersetzen eines C-Programms](#)

[UID|seeBenutzer-ID](#)

[ulimit](#)

[ulimit|textbf](#)

[umask-Shellkommando](#)

## Umbenennen

[einer Datei@Datei](#)

[von Kommandos](#)

[Umgebung|seeProzeßumgebung](#)

[Umleitung](#) , 

[umount](#)

[umsdos|seeDateisysteme](#)

[unalias-Shellkommando](#)

[uname](#)

[uncompress](#)

[uniq](#) , 

[unset-Shellkommando](#)

[Unterbrechung einer Schleife](#) , 

[Untergerätenummer](#) , 

[until-Schleifen](#)

[Usenet|](#)(

[Usenet|](#))

[UTC](#)

[utmp](#) , 

[uucico](#) , 

[UUCP](#) ,  , 

**UUCP**

[config-Datei](#)

[dial-Datei](#)

[Konfiguration](#)

[Konfigurationsdateien](#)

**Konfigurationsdateien**

[Format](#)

[Log-Dateien](#)

[port-Datei](#)

[Protokolle](#)

**Protokolle**

[g-Protokoll](#)

[weitere Protokolle](#)

[sys-Datei](#)

[Taylor-UUCP](#)

[uustat](#)

[uucppublic-Verzeichnis](#)

[uustat](#)

[uux](#)

[Valid-Flag](#)

[Verdoppeln einer Datei](#)

**vergleichen**

Dateien , 

Zeilen

verketteten v. Dateien

**Verschieben**

einer Datei@Datei

Verschlüsselung|

Verschlüsselung|)

versteckte Dateien

Verzeichnis

**Verzeichnis**

aktuelles

aktuelles anzeigen , 

anlegen , 

anzeigen

auf MS-DOS Disketten

auflisten , 

durchsuchen

Größe anzeigen

im Minix-Dateisystem

loeschen@löschen

Modus|

Modus|)

wechseln , 

Verzeichnisbaum

view|seeelvis

virtuelles Terminal

virtuelles Terminal|

virtuelles Terminal|)

vi|seeelvis

Vorzeichen

wait-Shellkommando

wall , 

wc

while-Schleifen

[who](#)

[Wildcards](#) ,  , 

[Wine](#)

[Woerter zaehlen@Wörter zählen](#)

[write](#)

[Wurzelverzeichnis](#)

[xfsck](#)

[xiafs|seeDateisysteme](#)

[xntpd](#)

[XON/XOFF](#)

[YXZ-Modem](#) , 

[Zahlensysteme](#)

[zcat](#) , 

[zdiff](#)

[Zeichen](#)

[zeichenorientiertes Gerät](#)

**Zeichensatz**

[ISO-Latin1](#)

[laden](#)

[Tabelle umschalten](#) , 

[vt100-Grafik](#)

[Zeichensatz|](#)(

[Zeichensatz|](#))

**Zeilen**

[numerieren](#)

[umbrechen](#)

[vergleichen](#)

**Zeit**

[Ausführung eines Kommandos](#) , 

[Datum](#)

[Zeitmarke einer Datei](#)

[Zeitzone](#) , 

[Zeit|](#)(

[Zeit|](#))

[zmore](#)

[Zombies](#)

[Zonenzeiger](#)

[Zufallszahlen](#)

[Zugriffsrechte](#) ,  ,  ,  , 

**Zugriffsrechte**

[ändern](#)

[Zugriffsrechte|](#)(

[Zugriffsrechte|](#))

[Zylinder](#)

Up: [Das Linux Anwenderhandbuch](#) Previous: [Index](#)

# Über dieses Dokument ...

This document was generated using the [LaTeX2HTML](#) translator Version 97.1 (release) (July 13th, 1997)

Copyright © 1993, 1994, 1995, 1996, 1997, [Nikos Drakos](#), Computer Based Learning Unit, University of Leeds.

The command line arguments were:

**latex2html** LHB.tex.

The translation was initiated by Sebastian Hetze on 3/15/1998

---

*Das Linux Anwenderhandbuch*  
(C) 1997 [LunetIX](#)

# Bestellung des Linux Anwenderhandbuches

Das Linux Anwenderhandbuch gibt es in gedruckter Form in jeder besseren Buchhandlung. Unter der ISBN 3-929764-06-7 kann es in jedem Fall leicht bestellt werden.

Sie haben auch die Möglichkeit, die gedruckte Version des Linux Anwenderhandbuches direkt von uns zu bekommen. Wir schicken Ihnen das Buch an dem der Bestellung folgenden Werktag per Post zu.

Die aktuelle 7. Auflage des Linuxhandbuches hat 640 Seiten und kostet 59,- DM. Bei einer normalen Postsendung berechnen wir zusätzlich 4,- DM für Porto und Verpackung, eine Nachnahmesendung kostet 8,- DM. Wenn Sie das Buch schriftlich bei uns bestellen und im voraus mit einem Verrechnungsscheck bezahlen, entfallen die Versandkosten.

---

Nachname:

Vorname:

Straße:

PLZ:

Ort:

eMail:

Telefon:

Menge:

Versand:      Post Nachnahme (8,-- DM Versandkosten)

---

Wenn Sie das Formular fertig ausgefüllt haben können Sie die Bestellung

Um die Eingabe mit einem leeren Formular zu wiederholen können Sie es